VRay
THE COMPLETE GUIDE

VRay

THE COMPLETE GUIDE

Francesco Legrenzi

Original title
VRay - THE COMPLETE GUIDE
by Francesco Legrenzi

Graphics and paging
Francesco Legrenzi
www.francescolegrenzi.com

First edition
2008

Printing
Industrie Grafiche Stilgraf
Via del canneto, 38/40
25010 Borgosatollo (BS), Italia

Printing finished during October 2008

ISBN 888813723-8

SUMMARY

CHAPTER 1 - Introduction

CHAPTER 2 - The complete story of Computer Graphics

CHAPTER 3 - VRay: Renderer PART 1

CHAPTER 4 - VRay: Renderer PART 2

CHAPTER 6 - CAMERAS AND LIGHTS

CHAPTER 7 - VRay: OBJECT and ENVIRONMENT

CHAPTER 8 - RENDER ELEMENTS

CHAPTER 9 - UPDATES

CHAPTER 1: INTRODUCTION

INTRODUCTION

GENERAL

ABOUT THE AUTHOR

Francesco Legrenzi was born in 1977 in a little valley in Bergamo (Italy). After earning a degree in building engineering at the faculty of engineering, he worked as a structural engineer for three years. Realizing his real passion, he abandoned the job for which he had spent years studying, to dedicate himself to the study and practice of professional graphic design. In 2001 he was asked to participate in the management of www.treddi.com, a small italian CG portal, that at the time counted up a bit more than 1000 users; at the present it is a point of refence for all italians fascinated by computer graphics, by January 2008 it had over 30.000 registered users. In 2001, he also became aware of a small plug-in, VRay, developed by Chaosgroup, a Bulgarian software-house. Since then he has not stopped being interested in everything that has to do with the world of rendering systems, especially VRay. After four years as the director of the Modeling & Lighting department in an architect's studio in Milan, the author is currently working freelance.

ACKNOWLEDGEMENTS

At the risk of sounding like as if giving an acceptance speech at the Oscar's ceremony, I thank the following people for giving me advice, inspiration, and support for this book:

- the friends and users of **www.treddi.com**. Without whose support and encouragement I would have never set off on this adventure.
- **Fabio Allamandri** (D@ve), who was with me for the drafting of the book.
- the users of the official VRay forum, the principal source of information and instruction. It is thanks to their advice that I have been able to make this book.
- the collegues at www.abc-fotografia.com, **Paolo Attivissimo**'s www.attissimo.net, **Mauro Boscarol**'s www.boscarol.com, **Francesco Banterle**'s www.banterle.com, for providing me with information key to the realization of the chapters dealing with lights, photography and the history of computer graphics. A thank you to **Paolo Bonavoglia**'s www.bonavoglia.eu for the part dedicated to photography, spazioinwind.libero.it/bravo/fotografia of **David Donnini**, web.tiscalinet.it/visionage/ of **Maurizio Chelucci** again for the part dedicated to cameras.
- **all the internet sites** where I have been able to find images and news to put together the chapter about the history of computer graphics, of light and of photography. I am sorry for not citing all the sources that contributed, even if only indirectly, to the drawing up of these chapters.
- the typographers **Stilgrafonline**.
- to **Mariarosa Tarantini** for her collaboration in the formal check of the text.
- to **Tom Hudson** for allowing the addition of the Greeble plug-in to the DVD.
- to **Cuneyt Ozdas** for allowing the insertion into the DVD of the Color Correct plug-in.
- to **Fedor Binka** for allowing the use of the GUI for vrimgexr program in the DVD.
- to the **Chaosgroup** for allowing the insertion of the VRay Demo into the DVD.
- to **Flipside** www.aversis.be/ for their HDRI maps.
- to **Chiara**, my girlfriend, for bearing my absence these years and for patiently encouraging me.
- to my dear parents **Tomaso** and **Megi**, who, without even knowing, taught their son commitment to follow his dreams, never letting go. Thank you!

PREFACE

Some years ago I had the opportunity to write some tutorials for *VRay*. They were pretty extensive, it's true, but I never imagined they would be so successful. After that, some users suggested the idea of putting a book together. At the beginning I didn't take what, at the moment, seemed like an absurd undertaking, seriously. As months passed, I wrote other tutorials, without publishing still, because it was just fun for me, not at all boring. I wrote so many that I ended up with a little booklet in my hands, and not only about *VRay*. And so I asked myself: "Why not think about the idea of making the book seriously?" I started jotting down some ideas, thinking about a table of contents and I understood that the idea was feasible! Initially I only wanted to deal with some aspects of *VRay*, writing a small manual of no more than 200 pages, but the more I wrote, the more I realized that I had many things to talk about. From a small project that would take a few months, putting the book together revealed itself to be much more complicated and intricate. After two years of work and almost 1.000 pages, I still look back with satisfaction at the efforts made to complete this difficult adventure. Naturally, difficult moments did not lack, during which I thought I would not make it, but in the end the thirst for knowledge and the passion for my work were victorious.
I've had to fight against, as a manner of speaking, the continuous updates of a program that has never stopped evolving. Starting in 2005, with the *VRay* v1.47.03, finishing in 2008, with the v1.50 FINAL. It was necessary to learn how to use them and to test them out at the same time, while dealing with all the problems that would arise. I had to create all the 3D scenes and the models (some were fortunately free). I had to create animations and all the 3,200 renders! Screen captures and composition of all the images, for a total of 2,300 images! To bring this volume to conclusion, I have worked whole nights, weekends and sultry holidays in the company of a little air conditioner. Still and all, I was able to finish the book.
I hope you find my work useful, because I put my heart and soul and all my expertise into this work.

Francesco Legrenzi
Milano.

ABOUT THE BOOK

This book teaches that the fabulous computer generated scenes, that look like real photographs to the naked eye, in the end, are not that difficult to create. The book is mainly addressed to users of *VRay* for 3ds Max. But it can also be used by all those that have the fortune of having the software for which the Rendering portware by Chaosgroup has been created. By now *VRay* exists for many programs, such as Cinema4D, Maya Sketch-Up!, Rhino and soon it will be available for XSI. *VRay* versions for other programs do not vary at all from the version for 3ds Max, because the core, that is the heart of the program, is always the same. The only thing that changes is the interface, depending on the software on which it is installed. The commands and their corresponding functions remain identical for all versions.
The aim of the book is to give a logical, complete, and meticulous explanation of **all the commands** in *VRay*, spread out for 3ds Max. Every now and then also tutorials will appear, in order to help understand parameters that are particularly complicated. The volume ended up being a comprehensive guide of all the possibilities offered by *VRay*. Although it is not specifically addressed to architects, animators or designers, some examples or techniques discussed could be useful for them.
This book is created for users that have medium understanding of 3ds Max or that are acquainted with 3D technology. Many arguments, in any case, are dealt with thoroughly as to give even those with little practical knowledge the particulars to enable them to understand the whole, or at least a part of it, that which is being explained. *VRay* is a simple program, but at the same time it is complex, even the most experienced users can find unexplored areas. It is also well designed and applies the same settings in many parts: it is logical and coherent, so it is not necessary to memorize uncountable exceptions to the rules.
A 3D beginner, though, should have a notion of it. and of artistic technique. The 3D graphic designer is fundamentally and artist, therefore he should know the fundamentals of color, design, photography and art direction. The instrument is the computer, therefore he should be able to navigate within its memory and carry out the operations outside of *VRay* and 3ds Max. 3D graphics, but above all rendering, is a painstaking discipline because it requires, both an artistic and a technical approach, with a good dose of patience and self-control. One must not be discouraged in moments of difficulty. All graphic designers continually face new problems that must be solved. While following this book and working with *VRay*, it is necessary to remember that in order to assimilate the concepts time and practice are needed. If you begin with small projects and experimenting: a plane and a teapot or a box with a whole for window. Before being able to work seriously on a project it is necessary to get a good idea of what *VRay* is capable of offering.

ORGANIZATION OF THE BOOK

Research was done to find a style for the book that was sober, but at the same time rich and pleasant to read. Composed of 9 chapters, each of them dealing with distinct and easily identifiable topics:

1. **INTRODUCTION.**
2. **THE HISTORY OF COMPUTER GRAPHICS.**
3. **RENDERER - PART A.**
4. **RENDERER - PART B.**
5. **MATERIAL EDITOR.**
6. **CAMERAS AND LIGHTS.**
7. **OBJECT & ENVIRONMENT.**
8. **RENDER ELEMENTS.**
9. **UPDATES.**

The first chapter contains general information. The second chapter is an unusual part of the book. Personally I've always had an interest in history, and that of Computer Graphics has, and I think many readers will agree, a special charm. For many years of my life I have collected articles and information about the subject. Realizing that I had gathered a lot of material, I decided to share it, by adding it as an integrated part of the book. Furthermore, in the same chapter, the history of *VRay* will be discussed, from its beginnings, with images and news never published before. Chapter 3 and Chapter 4, the largest ones, contain information regarding the use of most of the parameters in the Renderer. The subject has been divided into two parts only because of practical reasons. Chapter 5 describes all the material and maps added to 3ds Max after installing *VRay*, while Chapter 6 offers information about cameras and lights belonging to *VRay*. Chapter 7 describes the objects and the atmospherical effects made available by *VRay*. Chapter 8 introduces video compositing concepts, together with the Render Pass. The last Chapter brings the book up to date with the innovations introduced by *VRay*'s v1.5RC3, 1.5 FINAL, VRay 1.5 SP1 and SP2.

GET THE MOST OUT OF THE BOOK

Most techniques used in this book require only 3ds Max 8, 9, 2008 and *VRay*. For the creation of this volume many versions of *VRay* were used. While hopping from one version to the other, some codes and parameters have been added or removed. This is why it is very probable that some images do not correspond perfectly well with the interface of the latest version available. This is not a problem at all, because the concepts present are often the same, even if the commands have changed position.
To get the most out of this book, still, it is useful to have some experience in some 3D applications. For example, all graphic artists should have and know at least one painting and/or image retouching program. Programs such as Photoshop or Paint Shop Pro, Gimp or Painter are absolutely necessary to create and retouch the textures or the final renders.
As far as the reading of the book is concerned, the advice for newcomers is to begin with the first chapter, avoid jumping from one argument to the next. This is because often and deliberately, the description of some parameters referred to previously

THE ENCLOSED DVD

A DVD-ROM is enclosed to the book. It contains 6 folders, 5 of which are dedicated to Chapters 3, 4, 5, 7 and 9. Inside 32 animations in .mov format are present, in addition to many images taken from the book. The illustrations are divided as in the chapters and will have references in the text. The videos show some interesting technical situations. The remaining folder contains all *VRay* versions presently available. All DEMO versions for 3ds Max, Maya, Rhino and SketchUp are present, also 2 plug-in freeware, Color Correct, Greeble, and GUI4vrimg2exr, graphic interface for *VRay*'s vrimg2exr tool and the DEMO version of 3ds Max 2009.

WHAT VRay IS

VRay is a program that appeared in public in december 2001. It derives from the genius of two programmers from Sofia (Bulgaria): **Vladimir Koylazov** (Vlado) and **Peter Mitev** (Peter). *VRay* is an internal Plug-in present in 3ds Max. And it is used as a rendering engine substitute for the ones already present in the software. Lights generation, shadows, reflections, etc... are bypassed thus calculated by *VRay*. As the programmers announced, *VRay*'s strong point was the extreme rapidity in raytrace computation compared to traditional systems. Absolutely true! It was possible to render hundreds of crystal glasses in raytrace in an unimaginable time for Scanline, which was the only internal rendering engine available for 3ds Max. The Global Illumination too was very fast. Thanks to the innovative Irradiance Map renders with photorealistic quality in unimaginable times not possible before. Furthermore the interface, compared to the rivals, such as Brazil, finalRender and Mental Ray, was unquestionably clearer and more compact, and able to create stunning scenes. This wasn't all. Other effects like DOF, Motion blur, and the fastest existing sampling system were put into *VRay*. Moreover, it was possible to compute effects as caustics, and it contained special kinds of cameras. Last but not least the price, really competitive. All this caused a large growth in the user community. In few years hundreds of websites and specialized forums for *VRay* were created, amongst this ***www.treddi.com***.

VRay AND THE HARDWARE

Considering the high quality of the works achievable with *VRay*, one might think that a powerful system is necessary to use *VRay*. This is not true. Any reasonably recent computer is suitable for *VRay*. We could say the minimal requirements are those necessary to run 3ds Max.

SOFWARE

32-Bit Operating Systems:
- Microsoft Windows XP Professional (Service Pack 2 or higher) recommended.
- Microsoft Windows 2000 Professional (Service Pack 4).

64-Bit Operating Systems:
- Microsoft Windows XP Professional x64.
- Browser web required: Microsoft Internet Explorer 6 or higher.
- Additional software required: Direct X R 9.0c (required), OpenGL (optional).

HARDWARE

32-Bit Versions:
- Intel Pentium IV, Intel Core duo, AMD Athlon XP or higher processor.
- 512 MB RAM (1 GB recommended).
- Hardware accelerator OpenGL R and Direct 3D.
- Pointing disposal Microsoft Windows-compatible (Microsoft IntelliMouse).
- DVD player.

64-Bit Operating Systems:
- Intel EM64T, Intel Core Duo, Intel Xeon, AMD Athlon 64 or higher, AMD Opteron Processor.
- 1 GB RAM (4 GB recommended).
- Hardware accelerator OpenGL R and Direct3D.
- Pointing disposal Microsoft Windows-compatible (Microsoft IntelliMouse).
- DVD player.

Two versions of *VRay* exist: the 32-bit one, and the 64-bit one. Regarding the 32-bit version, it must be installed in the presence of a 32-bit OS, such as Windows XP Professional 32-bit, instead the 64-bit version must be installed with Windows XP Professional 64-bit. The 64-bit version will guarantee the possibility of assigning to 3ds Max, and to *VRay* as well, more than 4 GB of memory, due to its intrinsic nature. The impossibility of assigning more than 2 GB of memory to the same process is one of the most serious 32-bit systems problems. This is a big limitation, considering that high quality computations need a lot of RAM. In case the need exceeds the maximum memory available, 2GB in 32-bit systems, 3ds Max will crash, even if more than 2 GB are present on the system. This doesn't happen with a 64-bit system, where all the installed RAM, 2, 4, 8 or 16 GB, can be assigned to an unique process.

Some "not-official" advice on the typology of hardware to use with *VRay* to get the most out of the book are collected in the next section, with special care to the prices. This is not general advice: they're grounded on experience and on the present prices, according with *VRay*'s needs.

Processors: this is the part of the hardware which is used most by *VRay*. All raytracing computing, Global Illumination is processed by the CPU. Thus, the more power one has, the more rapidly the image creation will be achieved. Nowadays market offers relatively cheap 2.66 GHz-Quad-Core processors. This means the PC will pragmatically contain just one CPU which is in truth formed by 4 separated CPUs. A system containing two of these CPUs, called Dual Quad-Core, will halve times compared with the previous one. It is also possible to connect more PCs, Single Core or Multi Core, together in a network making them work together. This will allow a further fall in waiting times.

RAM: Memory quantity is another crucial aspect for a workstation. When *VRay* has to compute complex images, it stores the information in the system memory. Sometimes, when certain objects are used, for instance Light Cache, Displacement and many other ones, it could happen that the memory request by *VRay* is higher than the available one: this will make the program crash. This means it's better to economize on other hardware elements, rather than the RAM. As far as 32-bit systems are concerned, I find pointless to exceed 2 GB, while for 64-bit ones the advice is having at least 4 of them, divided in two benches, each one of 2 GB. RAM's rapidity is not important as its quantity.

HD: The choice of hard disk instead, compared to RAM and CPU, isn't all that important for *VRay*. Its employment is in fact purely storing the information. The faster a HD is, the more rapidly you'll be able to open a file, save or recall the Irradiance Map, etc...The differences between the new hard disks and the oldest ones is, anyway, just a few seconds. The problem arises when, for example, the HD is used by *VRay* as a "pad" memory, instead of the RAM. Sometimes you'll notice a HD's ongoing activity during the rendering with *VRay*, together with the processor's lethargic one. The reason is the RAM request has been integrated by the hard disk memory, achieving the creation of paging files on disk: this is a very long procedure. In other words, *VRay* is using disk space as if it was common RAM. This operation is very slow, because the hard disk is, for its nature, slower than the RAM. Having a fast HD could be useful for this operation, in slang called Swapping. The advice could be having a 250 GB HD, with rotation speed of 7200 rounds/minute.

Network Card: A good quality network card is helpful in case the PC is connected to a network used for data transfer thanks to a distributed render. The data is stored on a PC, when multiple machines work together to compute a scene. Once the render is started, this machine distributes the scene, the maps and textures etc... to all the other PCs linked to the network too. Once this process is finished, as to say all the PCs have received all the informations, the network traffic slows down quickly, but doesn't stop. A little activity, due to the continuous data transfer from the network computers, which are executing the render, to the central server, which receives the data deriving from the PCs, will always be present. Nowadays you can find network cards 10/100/1000 Mbps at a dozen of euros.

Video Card: The video card doesn't affect the rendering computing in any way. It is only useful for having a better management of the viewport, allowing a faster visualization and a better real-time texture representation. For this reason it won't be considered in this description.

BOOK'S WEBSITE

The author of the text (***www.francescolegrenzi.com***) and administrator of ***www.treddi.com*** have created, at the address ***http://www.treddi.com/forum/vray_la_guida.htm***, some pages of support for the book in order to interact with readers and when necessary provide additional material. In the site there will also be a forum which can be used exclusively by the owners of this book. One can access this part by clicking on the link in the publicitary page and inserting the password which is to be found on the back of the book's cover.
It will be a meeting point for exchanging opinions about everything concerning VRay and the subjects covered in the book.

The author will be pleased to offer you this support, accepting suggestions to improve the future editions of this book, but he is not available for consulting or specific problem solving concerning users' specific projects.

You can write to the author, both in Italian or in English, to this address:

vray_the_guide@treddi.com

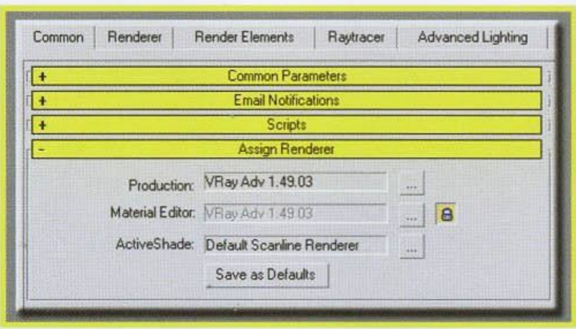
VRay AND ITS INTERFACE

VRay is a software built up to compute images by computer, specifically conceived to create photorealistic images. It is based on physical laws and properties present in nature. This enables it to simulate reality the best way possible thus obtaining physically correct renders.

■ **Figure 1.1**
Some examples realized by VRay's users.

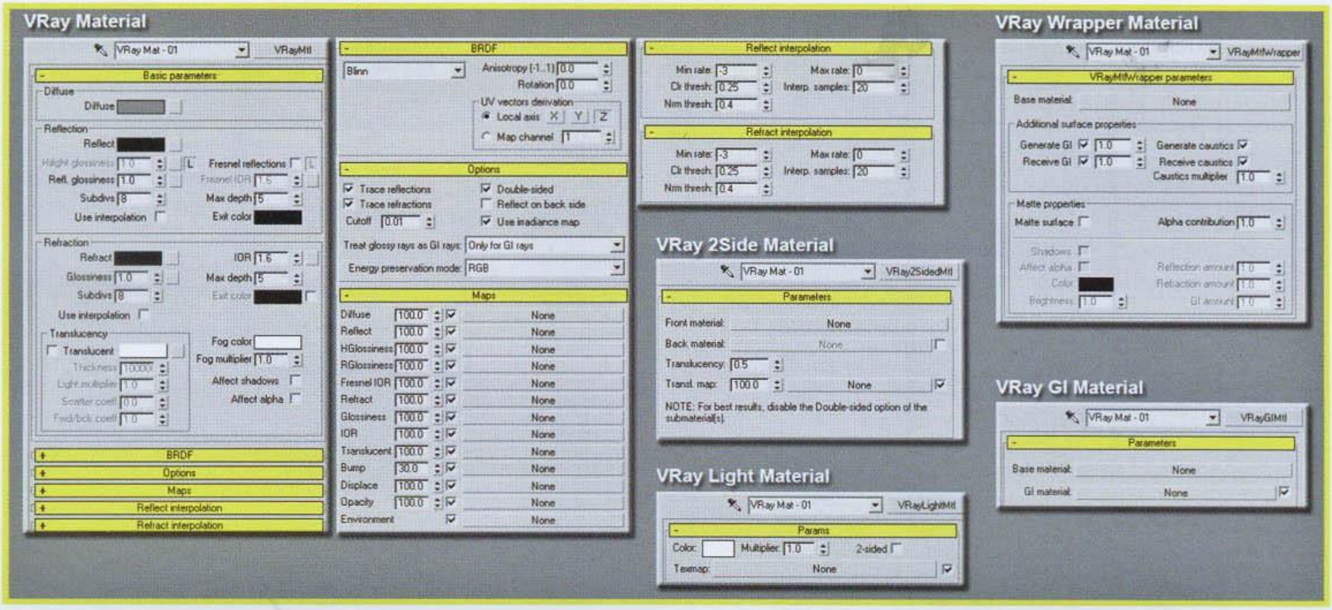
From the technical point of view, *VRay* was created as a 3ds Max plug-in. Nowadays many other versions are being developed, in the form of plug-ins for Maya, Rhino, Cinema4D, SketchUp, XSI, a Standalone version, one for Linux and another one for Mac OS X. The ones for Rhino, SketchUp, Cinema4D are already on the market.
We will examine only the 3ds Max version in this book. Since *VRay* is still under beta-testing, different kind of versions will be used, starting from ***v1.47.03***, up till the last version available during the book's writing. In case new features, deriving from new versions, become available, these will be remarked and commented.
(the book will be updated up to the ***v1.5 FINAL SP2*** for 3ds Max 2008).

We find *VRay* and its options spread inside 3ds Max. Once installed and set as rendering engine, one can find it in many different sections, such as Material Editor, or in the lights, the cameras, the objects, the effects, etc…

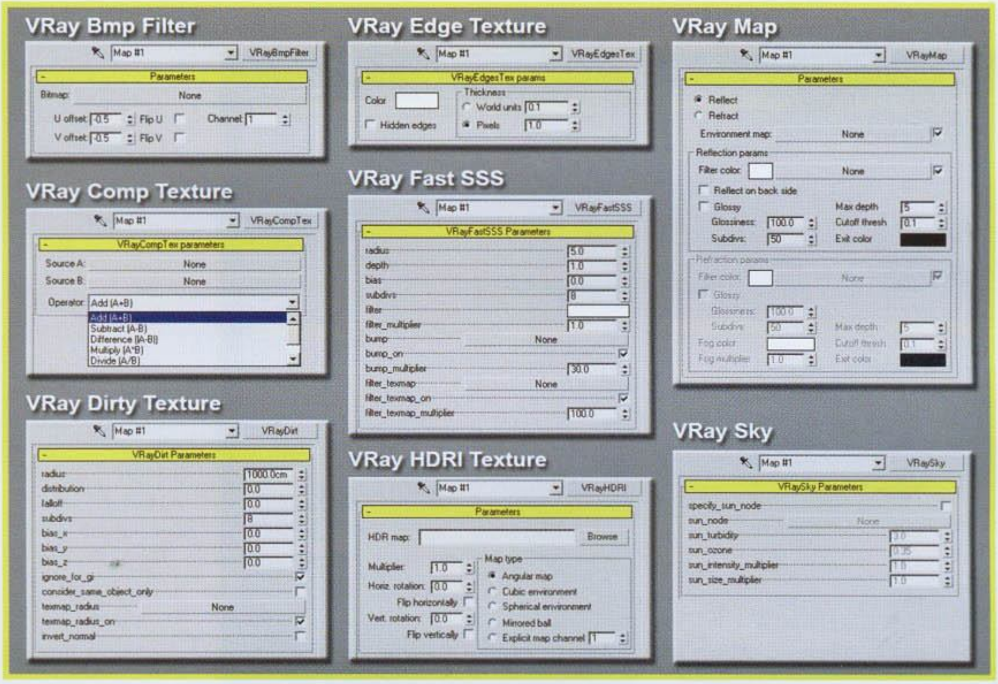


■ **Figure 1.2**
It is necessary to set VRay as a rendering engine to activate it inside 3ds Max. Furthermore, to make enable the Material Editor to manage itself new shaders, it is necessary to use VRay as the engine for the sample spheres render. If not, the spheres would be completely black and thus unusable.

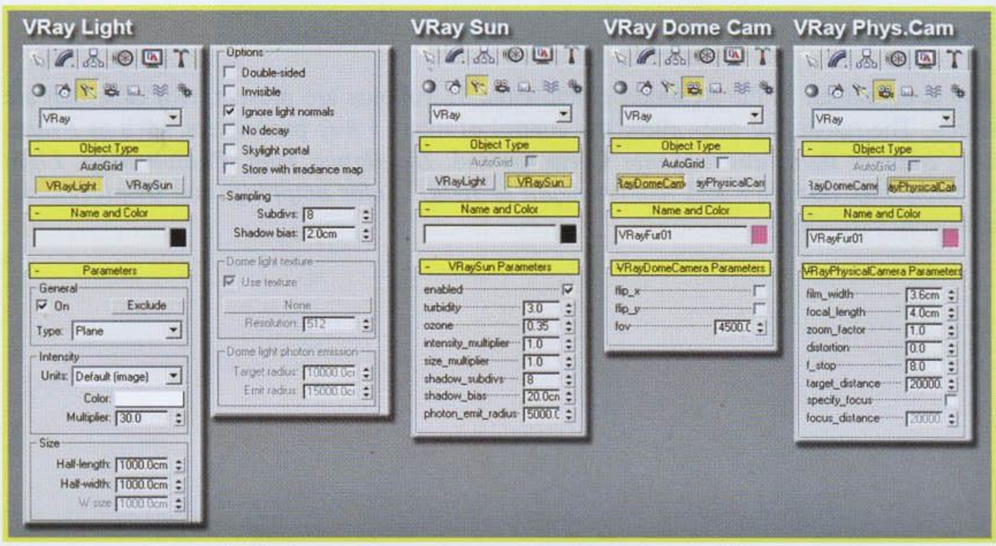
■ **Figure 1.3**
New shaders and maps present after VRay's installation.



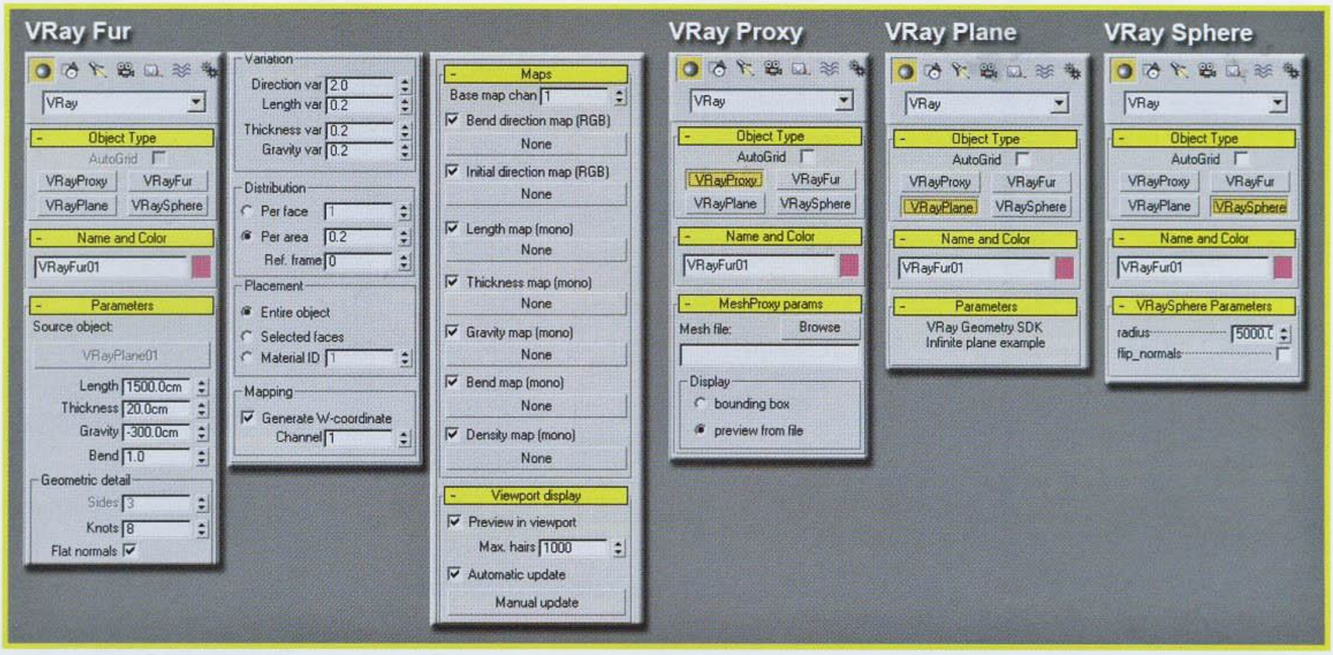
■ **Figure 1.4**
The five shaders available in VRay.



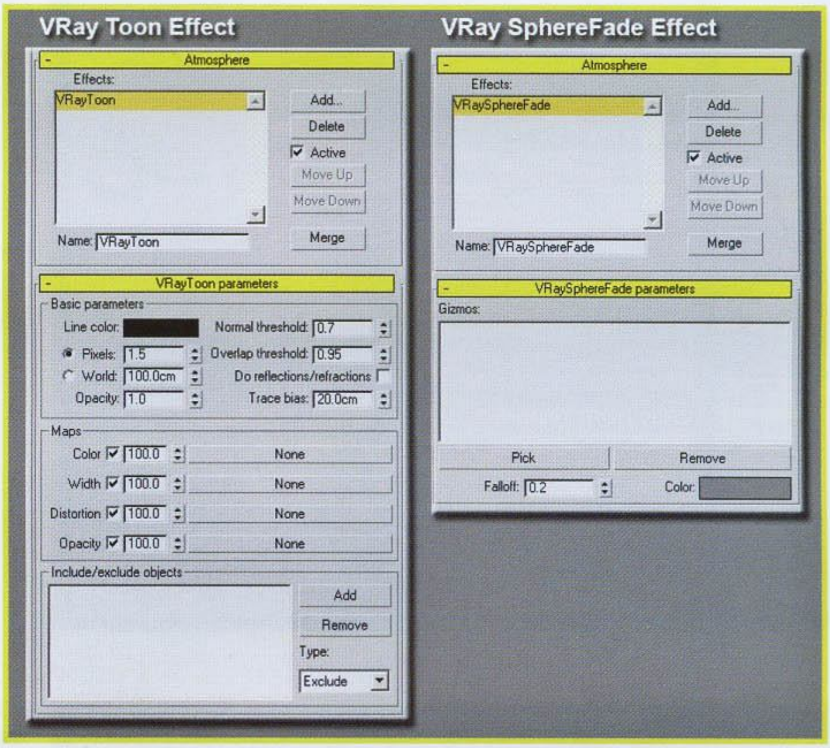
■ **Figure 1.5**
The eight new maps installed by VRay.



■ **Figure 1.6**
New lights sources and new cameras.

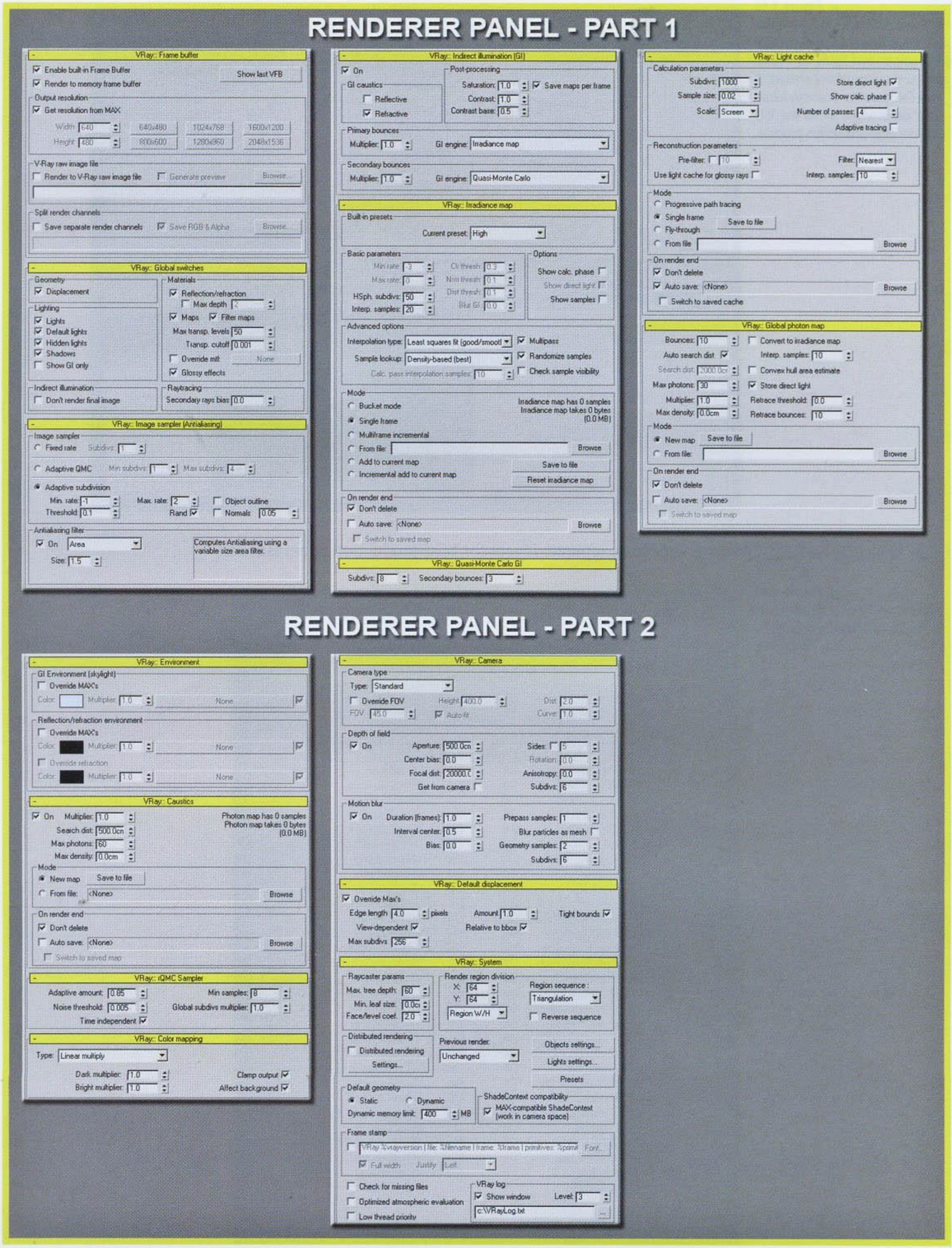


■ **Figure 1.7**
New ***VRay objects***.



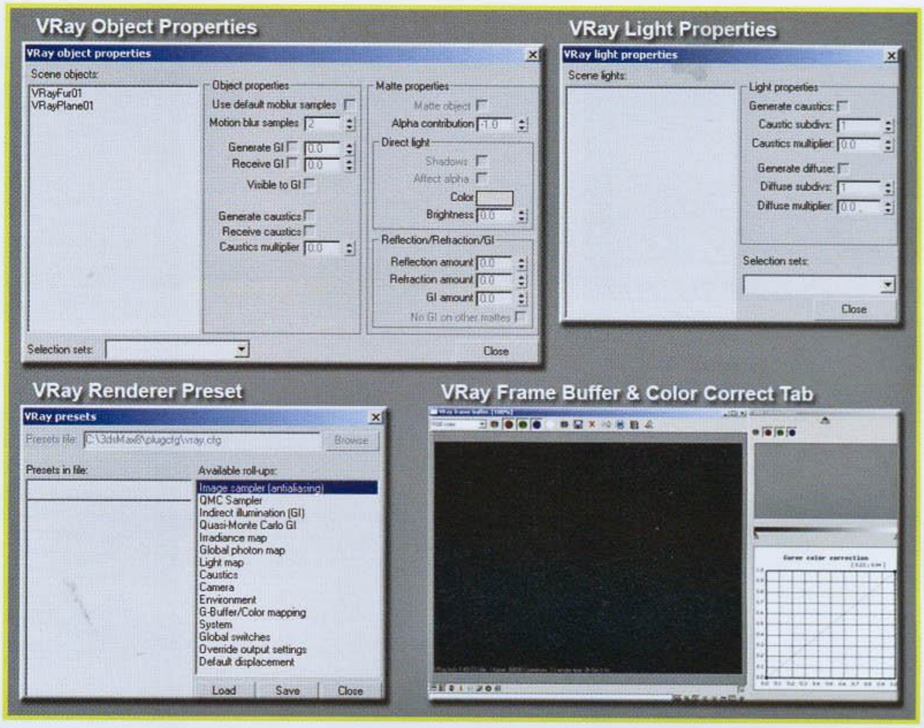
■ **Figure 1.8**
Two new VRay's effects: ***VRayToon***, for a "cartoon" effect on the rendering, and ***VRaySphereFade***, handy effect to speed up rendering's computing when objects on the move are present and not only.

The most important part of *VRay* is contained in the Renderer panel. Here 15 rollouts are present, two of which are devoted to logo and license loading. The remaining ones allow the control of many *VRay* parameters, such as ***Frame buffer*** management, ***antialiasing***, ***indirect light***, ***quasi-Monte Carlo*** method, ***Irradiance Map***, ***Photon Map***, ***Light Cache***, ***Environment***, ***Caustics***, ***Sampling***, ***camera's*** and relative ***depth of field***, ***Color mapping***, and ***Displacement***. The last rollout contains general system variables.



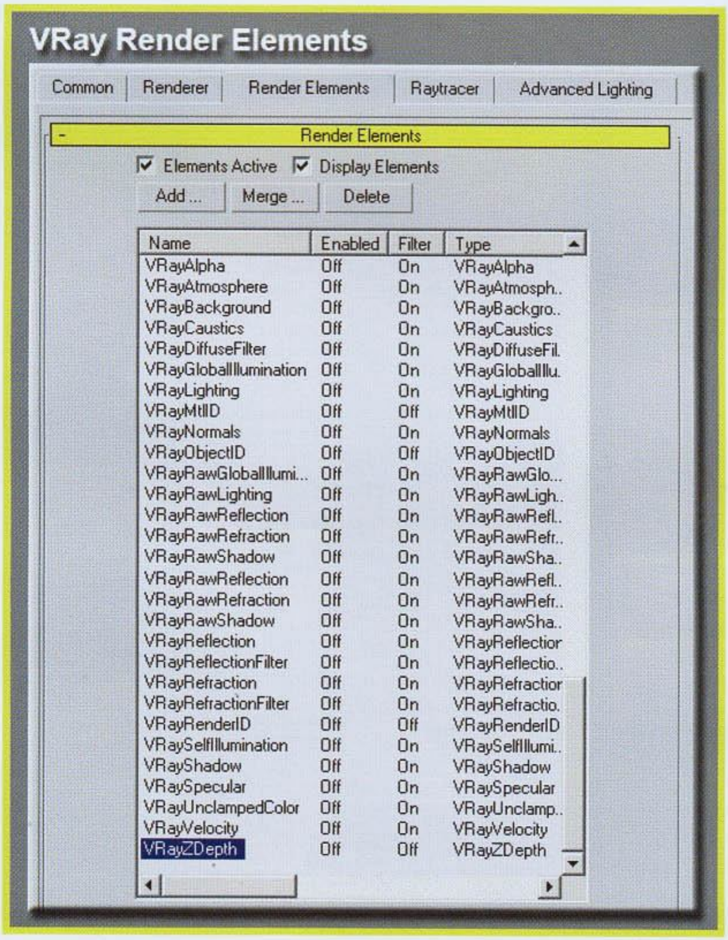
■ **Figure 1.9**
VRay's complex panel, the real heart of the program.

Within the Renderer panel other sub-panels are present, each one with specific functions.



■ **Figure 1.10**
Some dialogue windows which enables one to set parameters for single objects and lights, as well as save all the parameters present in the Renderer for possible future use; panel which allows the final render post-production process.

VRay backs 29 ***Render Elements*** usable as separate layers for post-production compositions.



■ **Figure 1.11**
The panel ***Render Elements*** of 3ds Max it is now supported by VRay too.

THE BOOK'S AIM AND ARRANGEMENT

The book is divided in chapters, following the same logic used in the previous images. Obviously the most important part is the one devoted to the Renderer panel, which will be subdivided in turn in two parts to achieve a better organization. In the next chapters, single objects, lights, cameras, etc... will be treated.
All single panels and rollouts will be treated as well, with detailed explanation and practical examples, to better understand the variations of each parameter. Wide space will be devoted to images and tests, which can explain a renderization program better than words.

CHAPTER 2: THE GREAT HISTORY OF CG

HISTORY OF COMPUTER GRAPHICS

GENERAL HISTORY

Which was the first computer in the world, who created it and why? What has led to a similar invention, which has become a main and prominent reality in the world we live in? When did CG arise and to whom was it aimed? Names like Catmull, Blinn, Phong and Photon Map are, nowadays, known, but where do they find their origin? What is Global Illumination? When was it born? Why is it so widespread? Who was the inventor of videogames? Everyone has wondered at least once in their life about these matters. In the next pages we will go across 2000 years of history, searching for the answers to these questions, resuming the main facts, knowing the characters, the inventions and discoveries that have led to the growth of this fascinating discipline.

1200 B.C.

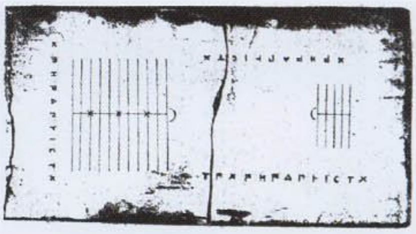
- Sumer civilization inhabitants wrote down commercial operations on **mud boards**.

■ **Figure 2.1**
Sumer's history has been pieced together thanks to fragmentary inscriptions in cuneiform alphabet found on clay boards and other archaeological evidence.

300 B.C.

- The oldest **counting board** was found on Salamis Island, belonging to Babylonians.

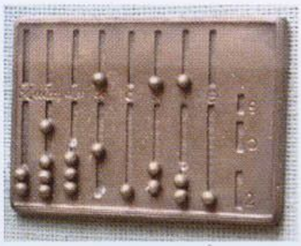
■ **Figure 2.2**
Salamina (Salamis in Greek), Eastern Greece island, located in the Saronic Gulf, a portion of the Aegean Sea, in the offing of Attica's coast, in the south-west direction from Athens.



100 A.D.

- **Calculus tables** in stone and metal, as those in this picture, are created during Roman Empire.

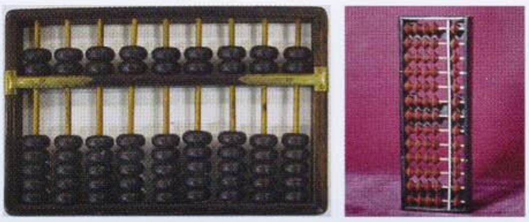
■ **Figure 2.3**
During the ancient Rome epoch some rudimental abacus were used. These were boards on which pebbles would slide.



1200 A.D.

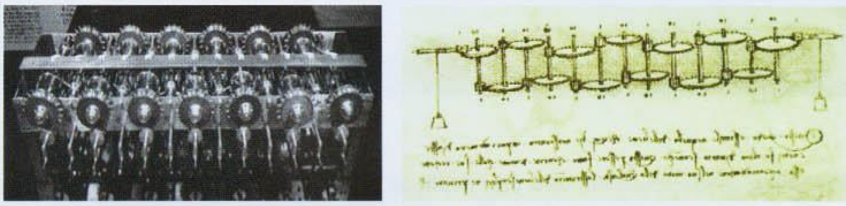
- The **abacus**, as we know it today, appears in China in 1220 b.C.

■ **Figure 2.4**
The chinese abacus, called suan-pa, is still widely used among populations in the Far East.



1500 A.D.

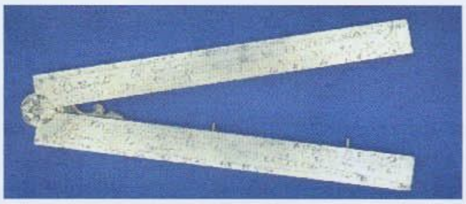
- **Leonardo da Vinci** plans, but doesn't achieve, the first **mechanical calculus machine**. But he doesn't know that, once built, it'll work.



■ **Figure 2.5**
- Leonardo da Vinci's model copy.
- Original mechanical design of the amounts to be carried.

1597 A.D.

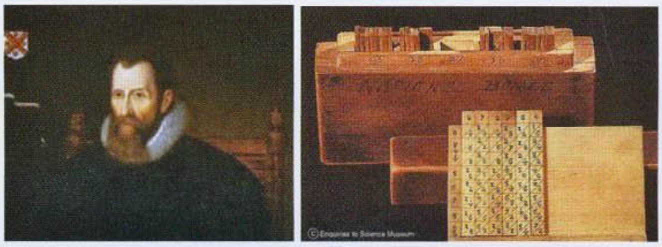
- In 1597 **Galileo Galilei** builds an instrument called **compasso geometrico et militare** (geometric and military compass), getting inspired by similar but rough instruments carried out by Niccolò Tartaglia from Brescia and Guido Monte. It is a sort of analogical calculator ruler, composed by two graded bars hinged together, thanks to which one can calculate squared and cubic roots an many other operations. Its use extends also to topography, surveying, and ballistic.



■ **Figure 2.6**
The scientist ordered about a hundred specimens from his mechanic from Padova (Italy). These were sold through the following years.

1613 A.D.

- **John Napier** (Nepero) invents **logarithms**, which will be the best instrument to simplify counts up till 20th century. The use of the comma to separate decimals is born. Napier uses numbered bars for the count.



■ **Figure 2.7**
Nepero's Tables. Multiplying, dividing, executing squared and cubic roots was possible.

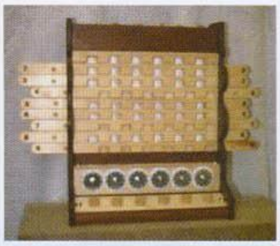
1623 A.D.

- **Edmund Gunter** creates the **first ruler** for logarithmic calculus.

■ **Figure 2.8**
Thanks to a compass multiplying and dividing was possible.

1623 A.D.

- **William Schickard** invents the **calculator clock**, able to automatically carry out additions, subtractions, multiplications and divisions.

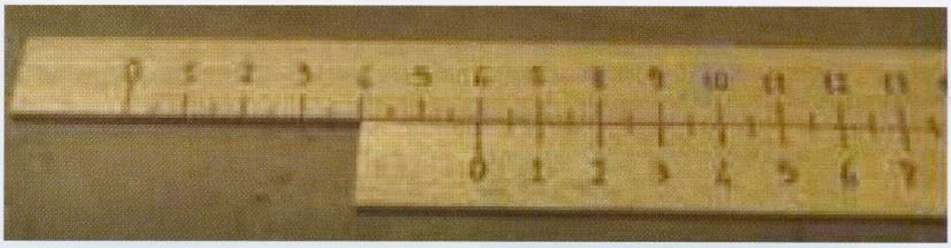


■ **Figure 2.9**
It used a clock's technology to execute sum and subtraction operations.

1650 A.D.

- **William Oughtred**, English mathematician, invents an elementary model of **linear calculator ruler**, based on the previous studies carried out by Nepero on logarithms and on **Edmund Gunter's** prototype. Letting the two rulers, on which the logarithms are written, slide one above the other, counting can be executed mechanically. Subsequently, thanks to the introduction of the third ruler and the cursor, this tool leads the way that will bring it to be the pocket calculator of entire generations of engineers, architects, mathematicians and physicists, up until the birth of electronic pocket-calculators.

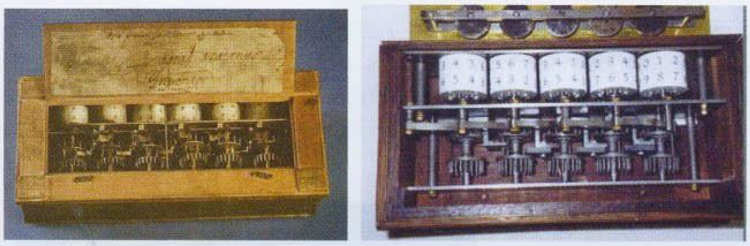
■ **Figure 2.10**
Beyond pocket models, also larger rulers will be created, up to 1 meter long and with greater approximations, to be used on work tables.



1643 A.D.

- **Blaise Pascal**, French philosopher, mathematician and physicist, at the age of 20, constructs a famous machine able to execute additions and subtractions automatically: the **Pascalina**.

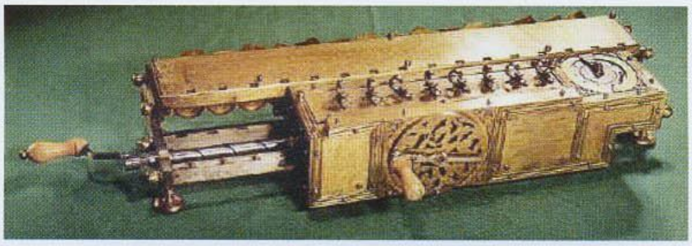
■ **Figure 2.11**
In truth a similar instrument, capable to execute multiplications and divisions, had already been fulfilled a few years before in Germany, but it was burned by a fire.



1674 A.D.

- **G. Leibniz**, German philosopher, mathematician and politician, discovers the **binary calculus**; unfortunately it falls into oblivion and it is discovered once again only in 1847, thanks to G. Boole, English mathematician who is to open up the path for the great schools of mathematical logic in the 20th century and most of all place a milestone leading towards the birth of the electronic calculator. In truth Leibniz bases his discovery on a calculus system introduced in China three thousand years ago that he bumped into during studies on the ideograms. Gottfried Leibniz creates a step calculator using a cylindrical mechanism.

■ **Figure 2.12**
Leibniz's machine reconstruction.



1714 A.D.

- **Henry Mill**, English inventor, patents the first documented attempt to build a **typewriter**.

1728 A.D.

- **Falcon** was a worker in a textile factory in Lyon. To speed up his work he invents the **Textile machine**. The **punched card** idea is introduced with his invention: the idea of a **program** as a pre-arranged instructions series too. Each weft thread is commanded, thanks to a set of metallic threads, by a crossbar located above the loom. A punched card series controls the weaving process. The punched cards correspond to a model to be reproduced: the presence or absence of a hole causes the warp thread's lifting or lowering. His invention was forgotten until 1801, when Jaquard, a mechanic, manages to create one to be used in factories.

■ **Figure 2.13**
Today, in textile manufacturing sector, the term "Jaquard" is used referring to a specific type of fabric.

1773 A.D.

- Pierre and **Henry Louis Jaquet Droz**, from Switzerland, invent the first **automaton** able to write. Shortly after they created another one, capable to design Louis XV's portrait. Generically, talking about automatons, we can actually say these have been the first robots in the history.

■ **Figure 2.14**
Even today Jaquet Droz's company works drawing inspiration from its famous founder, creating beautifully ornate clocks, characterized by care for details and manufacturing excellence.

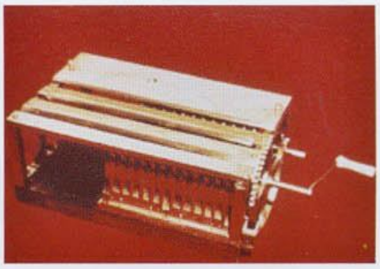
1774 A.D.

- **Philippe Matthaus Hahn** creates and sells few **calculators**, with 12 digits resolution.

■ **Figure 2.15**
Philippe Matthaus's machine.

1775 A.D.

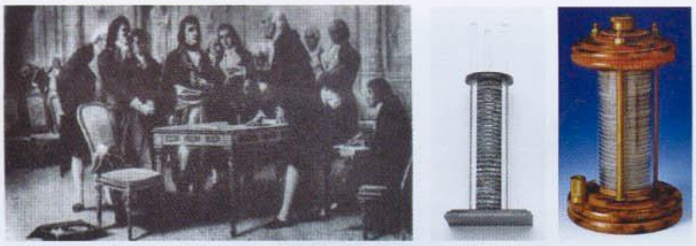
- **Charles Stanhope** develops a **calculator** capable of multiplying and dividing through a system based on multiple sums and subtractions.



■ **Figure 2.16**
Charles Stanhope's calculator.

1800 A.D.

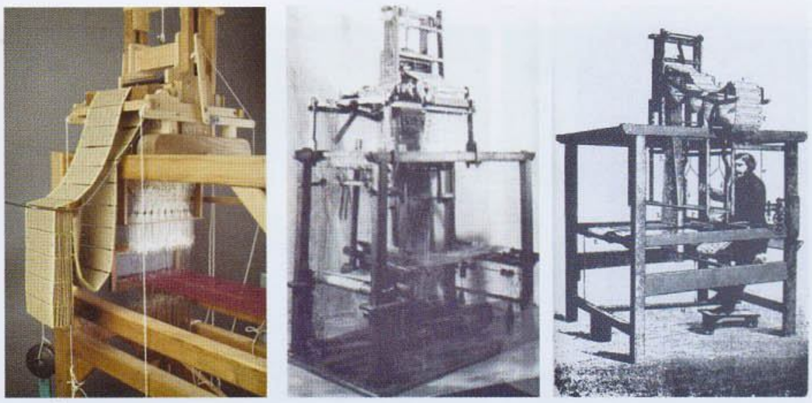
- **Alessandro Volta** manages to obtain electric energy, and succeeds in the objective of creating the **first battery** capable of giving electricity.



■ **Figure 2.17**
- A. Volta shows Napoleon an experiment.
- At the beginning he used separated cells for his experiments, connected in series; each cell was a wine cup filled with brine, in which two unlike electrodes were dipped.

1804 A.D.

- The first **punched cards** for the automatic running of the **Jaquard** loom come into play. The textile woof is commanded by the holes present on specific cards. More than 100,000 machines will be produced in the following ten years. The punched cards will be used for the looms until 1980.



■ **Figure 2.18**
Jaquard's invention was the cause of a real boom in textile industry and for this reason Napoleon gave him an honorary pension.

1811 A.D.

- The idea that machines could take work away from people starts to grow: protests and **boycott** movements born.

1812 A.D.

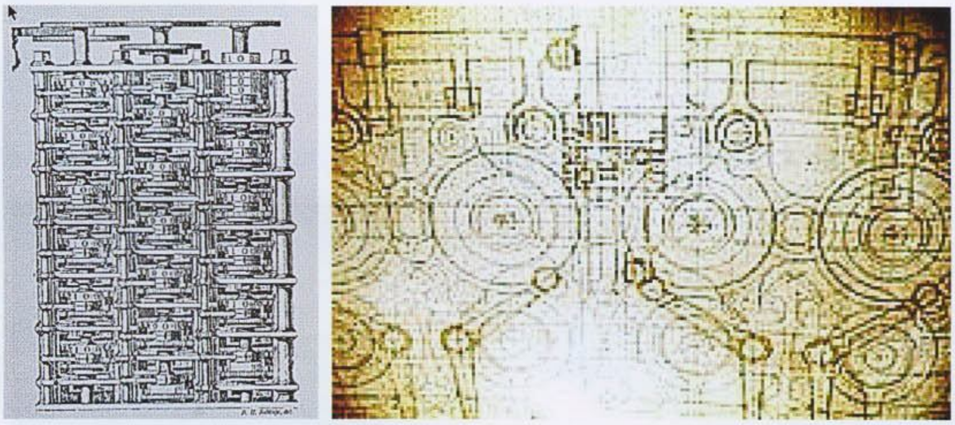
- **Charles Babbage**, a Cambridge mathematics professor in the UK, fulfils a lot of long and complex calculus, in particular those necessary for mathematic tables, are **composed by repetitive operations**, and thus are fit to be **automatically performed**. At that time teams of mathematicians would work night and day to create mathematics tables for logarithms and trigonometric functions. So Babbage thought he could create a machine to do that job. He starts the project of an automatic mechanic machine, which he calls **Difference Engine**.

Babbage knows about three kind of mistakes that could occur filling out the tables:

- human mistakes while calculating;
- copying mistakes during publication's transcription;
- mistakes concerning plates composition for the printing.

He thinks he'll be able to remove these mistakes projecting a machine able to cover the complete process, starting from the calculus going to typographical cliché. Contrary to what had been done until then, Babbage doesn't want to create calculating machines able to execute only the four basic calculations, but calculators conceptually similar to the modern ones, able to execute operations sequences in order with a program.

■ **Figure 2.19**
Drawing of the Original Difference Engine.



1820 A.D.

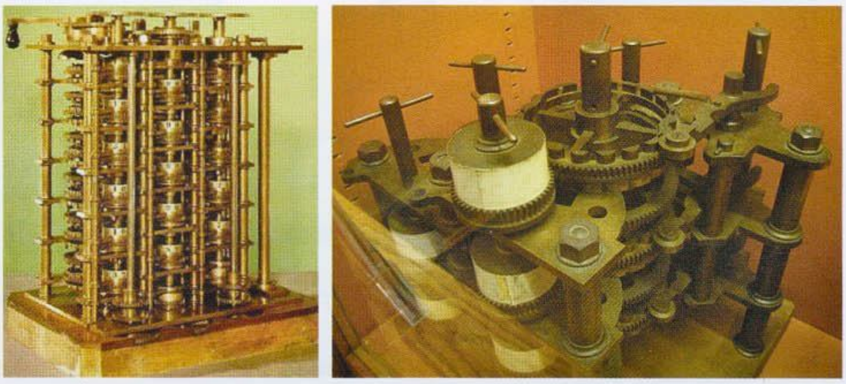
- **C.X. Thomas de Colmar** builds the **arithmetometer**, first industrially-used calculator able to execute the four basic operations without mistakes.

■ **Figure 2.20**
1,500 Machines have been produced in 30 years, and the production continued up till 1930 more or less.

1821 A.D.

- After ten years passed working, **Charles Babbage** achieves a demonstrative functioning model of his automatic calculator, or **Difference Engine**.

■ **Figure 2.21**
London's Science Museum creates a Difference Engine in 1990, perfectly working. Made in iron, steel and brass, it is composed by 4,000 components and it weighs more that 3 tons. In 1991 the device completes its first calculus loop, giving back a result exact to the 31rst decimal place. Each calculus involves the handles hundreds of times.



1822 A.D.

- **Charles Babbage** draws a new calculator. The idea was this machine had to work with punched cards, as Jaquard's ones and the results had to be printed on papers. Well, this was the revolutionary idea: **printing the results**! Anyway, Babbage believes this epoch's technology isn't sophisticated enough to fulfil his project.

In any case, a little scale model will be shown in an English Royal Astronomic Society meeting. The project will be abandoned in 1833, due to mechanics, pecuniary and opinion problems arisen between himself and his chief, an engineer, Joseph Clement. The concept of this machine can be considered extremely advanced, and probably too incomprehensible for the epoch.

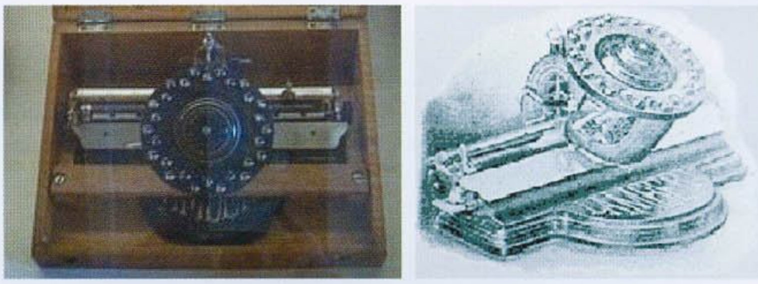
1826 A.D.

- **Joseph Nicéphore** invents a procedure based on the impression on a pewter plate, covered by bitumen, exposed in a darkroom. **Photography** is born.

■ **Figure 2.22**
An heliography fulfilled by Nicéphore in 1828. It has been obtained with a pewter plate, covered by bitumen coming from Judea, a substance capable of fading and harden when exposed to the light, while it remains soluble when in dark. The impression of the image requires many hours.

1829 A.D.

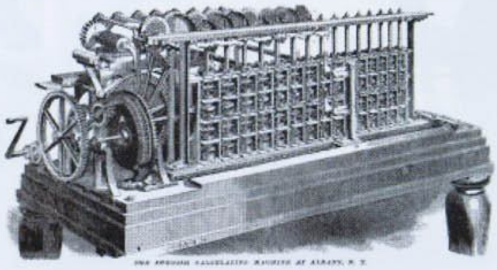
- **William Austin Burt**, American inventor, patents a new **typewriter**: his device is composed of a circular sector, on which a set of characters is arranged. The circular sector is rotated up till the desired character, and then pressed on the paper.



■ **Figure 2.23**
The first typewriter.

1832 A.D.

- **Georg and Edvard Scheutz**, Swedish father and son, try to develop a **Difference Machine** by themselves after having read Babbage's article. They are the first ones to have created a machine capable of printing calculated tables results too. The numbers are printed on paper or metallic sheets, ready to do the cliché for the reproduction of tables.



■ **Figure 2.24**
Their machine is completed in 1843.

1832 A.D.

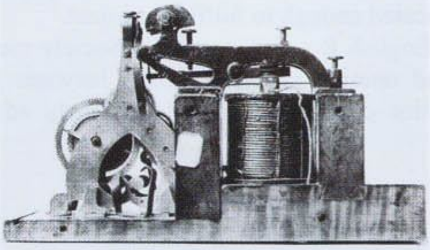
- **Xavier Progin**, French inventor, patents a **typeprinter** containing a modern characteristic: a single lever-key for each character or symbol.

1834 A.D.

- In this year **Charles Babbage** projects the **Analitycal Engine**, a faster and more accurate machine compared with the previous one. It is the first **automatic calculator model**, which will remain just a project, because of problems in producing all mechanical components needed and the low interest from possible financers. On the other hand Babbage spent a big part of his money to bring forth his challenging project. **Analytical Engine**'s idea was making use of Jaquard's punched cards cycles, in order to automatically control the calculations. In this way it was able to decide in order with the previous calculations. Some texts also tell about the idea of a steam engine to be applied to the analytical machine.

1837 A.D.

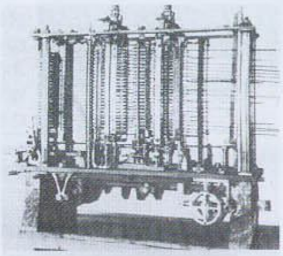
- **Samuel Finley Breese Morse** patents an efficient communication system: the **telegraph**.



■ **Figure 2.25**
The telegraph shown here was built by Morse, and used in 1844 to send the first public telegram, by using an apposite code: morse code. First telegraphs were able to carry messages roughly 30 kilometers away.

1843 A.D.

- **Lady Ada Lovelace**, Lord Byron's daughter, studies Babbage's Analytical Engine. She gathers from it the ideas of **loop** and **subprogram**, that is a repetitive steps sequence. A rich correspondence starts between them, adding to his job the idea of a machine able to operate via program. Lady Ada Lovelace is so considered as **the first programmer**.



■ **Figure 2.26**
A programming language will be devoted to Lady Ada Lovelace between the years 1970/80.

1846 A.D.

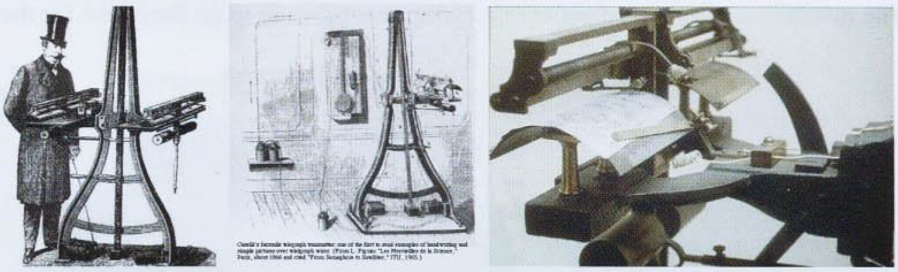
- **Alexander Bain** uses a **punched tape** to send telegrams. This system makes the transmissions faster and will be used up till the 20th century.

1847 A.D.

- **George Boole** writes **"An investigation on the Law of Thought"**. Here we find the relations between mathematics and logic, on which the **Boolean Algebra**, used for calculators circuits, is grounded. This causes a break with traditional mathematics, showing that logic is part of mathematics and philosophy.
Before this **AND, OR, NOT** concepts weren't applied to mathematics.

1861 A.D.

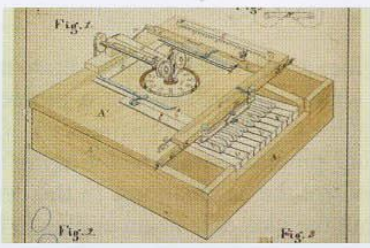
- **Giovanni Casselli** creates the first Fax machine, the **Pantelegraph**.



■ **Figure 2.27**
The Pantelegraph.

1867 A.D.

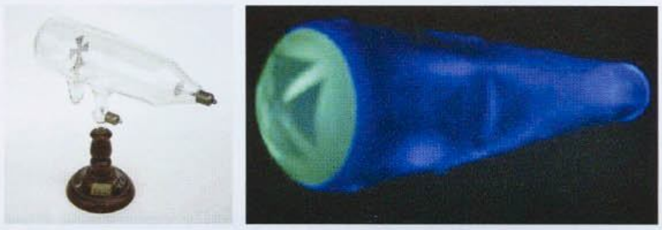
- The **typewriter** as we know it is born. It is one of those great inventions that will radically change comunications and administrations. In a few years manual writing will disappear. The QWERTY keyboard is born as well.



■ **Figure 2.28**
The first typewriter was created by Cristopher Sholes.

1870 A.D.

- **William Crookes** builds a modern **television tube** forerunner aimed at studying cathodic rays properties. Once created a vacuum condition and applied a high voltage, luminescence is produced on the end of the tube hit by cathodic rays. Little objects put inside the tube project a sharp-cut shadow on a luminescent wall.



■ **Figure 2.29**
Crookes gathers from this experiment the concept that cathodic rays move in straight lines.

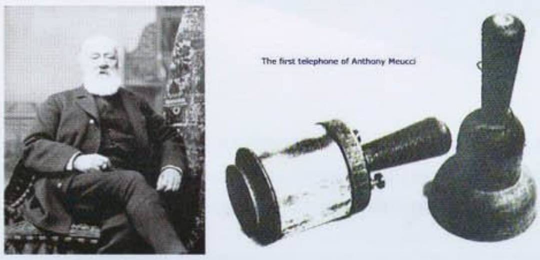
1870 A.D.

- **Electromagnetic field**'s theory is born thanks to **Maxwell James Clerk**, one of the most important scientists in 19th century. He assumes that light is an electromagnetic wave, moving at the speed of light. This hypothesis will be confirmed by Hertz more or less twenty years later. His ideas radically change the interpretation of luminous phenomena, paving the way to fundamental modern physics theories, such as relativity and quantum theory.

■ **Figure 2.30**
One could say Maxwell's theory gave the elements to Einstein to imagine another reality. From his name a famous rendering engine developed in 2005 will come.

1871 A.D.

- In 1857 **Antonio Meucci** invents the **telephone**. Meucci strenuously obtains a biennal patent and shows his invention's drawings to Western Telegraph CO's director after over a decade spent in the search for funds. Cheated in a despicable way because of the papers and blocked by his indigence, his invention is used by the professor Graham Bell, which gets all his merits and gains committing perjury. Meucci's fatherhood will be admitted late by the Supreme Court in 1886, because of famous and complicated actions brought from Bell's rivals. But Meucci will end in indigence his days in a house in Long Island, given to him by private American citizen.



■ **Figure 2.31**
Due to the lack of money, he couldn't have a normal patent for his device capable to electrically transfer the voice. Thus he was forced to accept a patent to be renewed each year for 10 dollars. He was asked to pay 200 dollars for a definitive patent, but he managed to scrape together just 20 dollars.

1876 A.D.

- Lars Megnus Ericsson founds the company **Ericsson**.

■ **Figure 2.32**
Mr. Ericsson.

1876 A.D.

- **Graham Bell**, of Scottish origin, asks for a patent for a telephone system. He uses Meucci's idea, founding the **Bell Telephone Company**.

■ **Figure 2.33**
Bell had studied the possibility of impression a magnetic field on a support to be later reproduced during his working period in Volta's laboratories. Because of his lacking knowledge this remained just an idea.

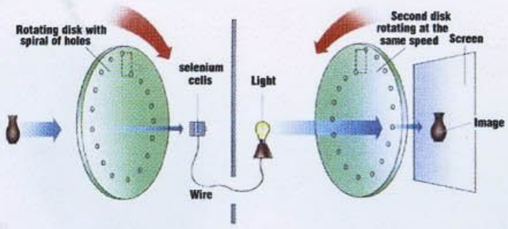
1878 A.D.

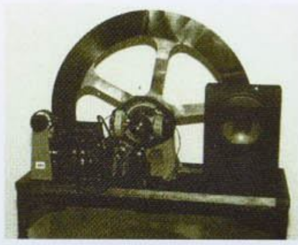
- **Remington** typewriter Standard 2. The first **typewriter** industrially produced and sold.

■ **Figure 2.34**
Remington Standard 2 typewriter's keys were already disposed in the modern QWERTY sequence, and this became a standard.

1884 A.D.

- **Paul Gottlieb Nipkow** invents the homonym disc. It is essentially composed by a pair of discs rotating with same frequency, on which there are a few small holes arranged in order to form a spiral originating from the center; when the first disc rotates, luminosity of the points of an image passes progressively from the holes situated on the surface to a photoelectric cell present in the space between the two discs. Very complicated to describe as well as to make it work. The first television, created by John Logie Baird, Scottish engineer, and called Radiovision, will be fulfilled many years after, using this system.

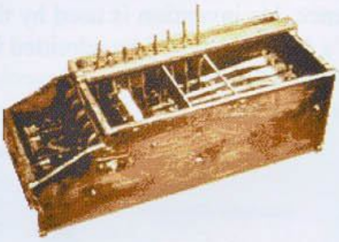




■ **Figure 2.35**
Functioning scheme and real specimen.

1885 A.D.

- **Dorr E. Felt** builds a calculating machine called **Macaroni Box**. From this the **Comptometer** will take origin, being competitive up until 1902.



■ **Figure 2.36**
The name Macaroni Box derives from the box in which the machine is contained.

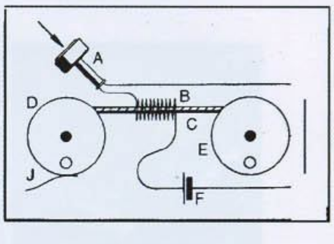
1888 A.D.

- **Thomas Alva Edison**, American inventor, introduces the **Kinetoscope**. This machine allows an observer to see moving pictures while looking into a hole (afterwards the term "moving pictures" will mean cinema). The machine will greatly work in "penny arcades" and country fairs. Inserting a coin in the machine the animation can be observed.

■ **Figure 2.37**
Over 1000 kinetoscopes were created in three years and roughly 250 movies were to be seen.

1888 A.D.

- **Oberlin Smith**, in an article appeared on the Electrical World, issues the possibility of **recording informations thanks to magnetization** on an apposite support.



■ **Figure 2.38**
Functioning scheme.

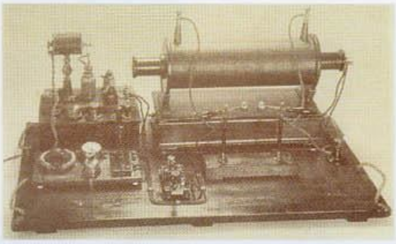
1890 A.D.

- **Herman Hollerith** beats the competition and is selected for the USA census in 1890, with his **Tabulation Electric System**.

■ **Figure 2.39**
Punched cards system is to be slightly modified in the following fifty years.

1895 A.D.

- **Guglielmo Marconi** transmits his first signal via **radio**. In 1896 the Italian scientist will be able to transmit signal over 1,500 metres away and the following year he will propagate signals from mainland to a ship located roughly 30 km far from the coast. In 1899 he will be the first to establish commercial relations between France and England; at the beginning of 1901 he manages to send signals 322 km far, and in the same year, he is able to send the first signal over the Atlantic Ocean.



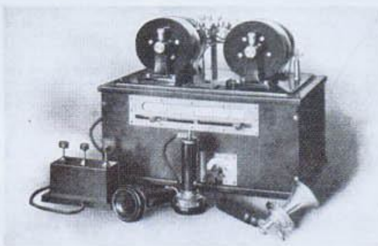
■ **Figure 2.40**
In 1902 messages were already regularly sent via radio through the Atlantic Ocean, and in 1905 on many ships radios were used to comunicate with coast stations.

1897 A.D.

- **Karl Ferdinad Braun**, German physicist, creates the first **oscilloscope**. Studies on the first cathodic tube will be based on this invention.

1898 A.D.

- **Valdemar Poulsen** patents the **Telegraphone**, first magnetic recording device.



■ **Figure 2.41**
The device is able to record and reproduce human voice.

1900 A.D.

- The appearance of the calculator in the job domain can be brought back up to the beginning of 20th century. Calculating machines produced at the beginning of the century were based on data processing technology. In fact, within big organizations data processing centres were present, equipped with a wide variety of machines, among which:

- a punching machine to make documents become punched cards thanks to Hollerith's code;
- a checking machine controlling the punching machine's job's quality:
- a selecting machine to arrange words in alphabetical or numerically order the cards;
- a calculating machine to execute numerical operations on the data read by punched cards or to punch the results on other cards;
- a tabulating machine to print the results.

Executing calculations was quite rapid at that time, roughly **60 operations per minute**. But their management was complicated, as you can imagine by the previous description. Furthermore these machines were able to execute operations on data series, something still far from what we're used to see today in modern computers.
This technology evolved quite slowly up to electromechanical machines, around 1940, which gathered different data processing components functional characters. The idea of a **program** as a sequence of instructions to be performed in order appears once again in these machines. Instructions were given in the form of punched codes on a paper band, similarly to Babbage's machine. In this case we talk about "outer program" electromechanical calculators.

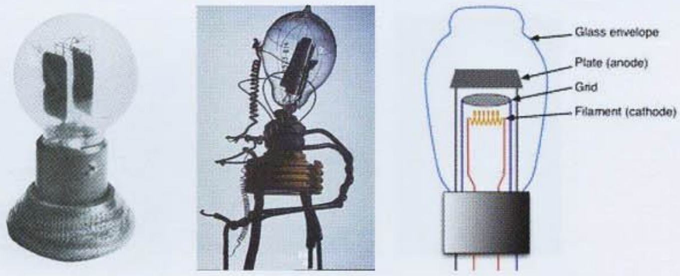
1900 A.D.

- **Max Planck** describes **quantum** effects.

■ **Figure 2.42**
This theory will be very important for microprocessors development.

1904 A.D.

- **Sir John A. Fleming**, English engineer, patents the **diode valve in vacuum**. This improves radio comunications. It is a glass shell, in which vacuum is created, containing a metallic filament brought to incandescence, between 1,000 and 3,000 Celsius degrees, by making electricity pass inside. The difference compared to the bulb is that the valve contains over one metallic elements, shaped into a grid or a screen, that can be connected from outside. The metallic filament, or best the metallic tube that winds it, is called cathode, while the metallic element outside anode.



■ **Figure 2.43**
The thermoionic tube is similar to the well-known bulb.

1905 A.D.

- **Albert Einstein**, German physicist, describes his **relativity theory**.

■ **Figure 2.44**
He himself attributed the development of his theory to his mental slowness. Another theory, more recent, concerning his mental condition, is that he was affected by Asperger's syndrome, similar to autism.

1906 A.D.

- **Lee de Forest**, American, adds the third electrode to Flaming's diode, the grid, creating thus the **triode**. Without this invention projecting digital electronic processors would have been impossible.

■ **Figure 2.45**
In 1904 he had patented the phonofilm also, first example of talking films. The sounds were recorded thanks to a luminous ray. His films have been shown, with recorded sound, in 1932 in New York.

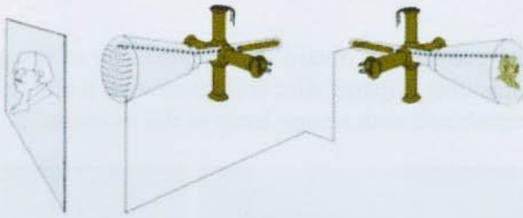
1908 A.D.

- Camillo Olivetti founds in Ivrea **Olivetti SpA**, Italian company working in the field of office and telecomunications systems.

■ **Figure 2.46**
Adriano Olivetti surely is the most unusual Italian industrialist in the 20th century. With socialist ideas, inheriting his father's factory, he made it become a breeding ground for industrial testing.

1908 A.D.

- **Campbell Swinton**, English scientist, describes an electronic scanning method and elaborates a way to compose images by using **cathode rays**.



■ **Figure 2.47**
Video signals transmission and reception.

1909 A.D.

- The forerunner of digital portable calculators is **Comptator**, invented in Germany around 1909 by **Hans Sabielny**.

■ **Figure 2.48**
Over 20.000 specimens were produced.

1911 A.D.

- **Hollerith**'s company - Tabulating Machine Company - merges with other two companies, creating the **Computing Tabulating Recording Company**, which will become, in 1924, the **International Business Machine Corporation**, that is to say IBM.

CTRCo	BUSINESS INTERNATIONAL MACHINES	IBM	IBM
Logo de CTR (1914)	Logo de IBM (1924)	Logo de IBM (1956)	Logo de IBM (Actualidad)

■ **Figure 2.49**
CTRC doesn't only produce calculators but balances and clocks too.

1912 A.D.

- Between 1873 and 1912, the American **Frank Stephen Baldwin** creates some calculating machine models based on the principle of thorn rack. The first successfully commercial operation happens only when he founds, in New Jersey, the **Monroe Calculating Machine Co** together with **Jay Randolph Monroe**, with whom he had also worked to fulfil a keyboard to be added to the machine.

■ **Figure 2.50**
Ten years later Monroe will be considered the pioneer of electro-mechanic calculating machines.

1920 A.D.

- The first **cash register** able to print numbers is born. CTR, becoming soon IBM, brings it into the market.

1923 A.D.

- **Vladimir K. Zworykin**, Russian physicist emigrated to the USA, achieves, for the Westinghouse Electric Corporation, a system for electronically filming images: the **iconoscope**, and a television receiver: the **kinescope**. Afterwards he will be supported by RCA which will fund the research and mass-produce the first electronic tubes.

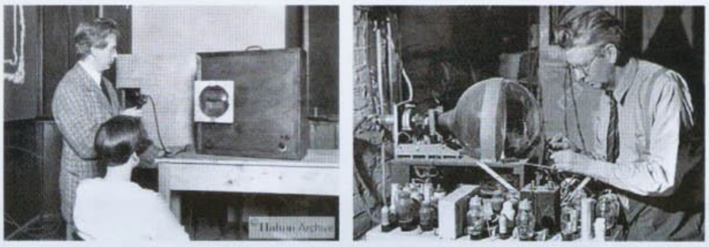
1924 A.D.

- Thomas J. Watson Senior renames **IBM** the old CTR company, with his popular writing THINK. He had already invented this slogan for the National Cash Register. This word will be written everywhere in IBM documents, and will last the following fifty years.

1926 A.D.

- In January 1926 **J.L. Baird** manages to obtain the transmission of images at a distance. To do this he uses a device composed by a mechanical scanning Nipkow disc combined with a photoelectric cell used to modulate the carrier wave and a similar syncrone disc combined with a neon lamp in the receiver.

■ **Figure 2.51**
This system, briefly used for experimental public transmissions, was surpassed later by Zworykin's iconoscope.



1927 A.D.

- The American **Philo Farnsworth** develops the first entirely electronic television system. He gives the patent to the RCA: **the television is born**.

■ **Figure 2.52**
Very talented, when he was 14 years old he had already imagined how to form images by using a beam composed by electrons covering many horizontal lines in succession.

1927 A.D.

- **H.J. Zeeman**, coming from the Netherlands, discovers that **silicon** behaves similarly to metal. Later on, in 1930, he'll discover silicon is a semi-conductor. Silicon will be used in all the chips built from 1954 on.

1928 A.D.

- Paul V. Galvin and his brother Joseph E. Galvin found **Motorola, Inc**.

■ **Figure 2.53**
The original name that is to say when it has been founded, was Galvin Manufacturing Corporation. It has been changed in Motorola in 1947.

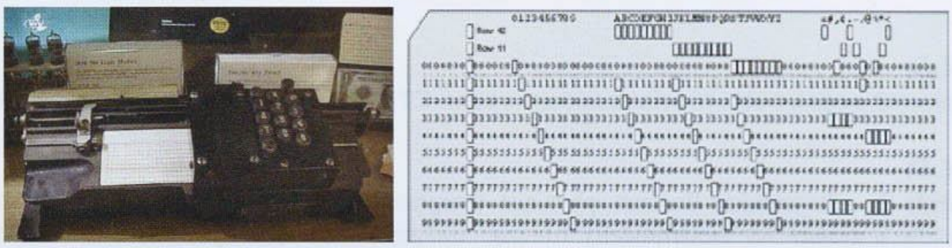
1928 A.D.

- Measuring time with **quartz crystal** makes an unimaginable resolution become possible.

1928 A.D.

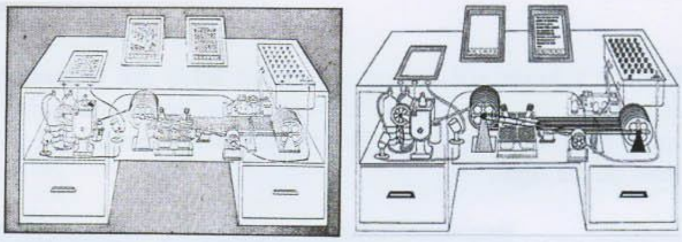
- **Fritz Pleumer**, of German origin, patents his **magnetic tape**. Data can be recorded and read. This invention is based on the famous magnetic thread, invented by Vlademar Poulsen in 1898. In the same year, the punched cards pass from having 40 holes to 80 holes. The **80 columns card**, used by **IBM**, becomes an industrial standard for a long time.

■ **Figure 2.54**
The first IBM punching machine. Its cards had the dimensions of a one-dollar note.



1930 A.D.

- **Vannevar Bush's** integrator-analyser: it is the first practical electronically functioning **analog calculator** to be used. Memex is born. This machine enables the solution of some differential equations. It is a sort of desk thought to make documents and materials concerning anything has been transferred in microfilm immediately available for the operator. Memex was described as a desk equipped with translucent screens, a keyboard, a set of buttons and levers. Within it motorized mechanisms for an ultra-rapid research in a wide microfilm archive, storing any type of materials to be printed, texts and images too, were present. Memex had to allow the operator to add marginal notes too, besides searching informations.



■ **Figure 2.55**
The most important thing Memex had to be able to do was creating stable links among different documents. This was possible by selecting the documents and pressing a specific button. This particular characteristic makes it the forerunner of hypertextual systems, like those which today make up the Web.

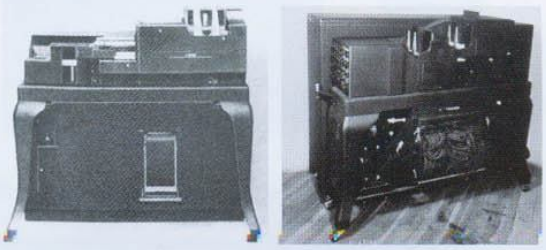
1932 A.D.

- **G. Taushek**, Austrian, invents the **Magnetic Drum**, basing this invention on Pleumer's priciples. He places a ferromagnetic plate on a rotating metallic cylinder. Some read and writing heads are positioned at some millimeters of distance one from another, and produce electromagnetic pulses. These pulses can be recorded on the plate, changing iron particles magnetic orientation. A drum with dimensions 20 cm (length) and 10 cm (diameter) has a capacity of 500,000 bits.

■ **Figure 2.56**
The first hard-disk, in its primitive appearance.

1932 A.D.

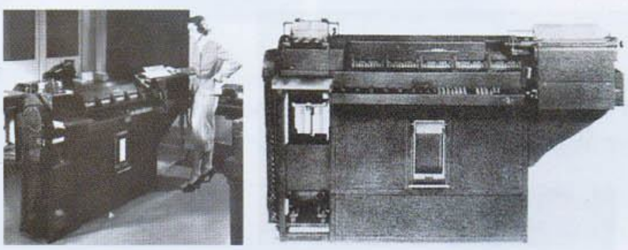
- **IBM** presents the **multiplier 601**: it reads two factors, with 8 decimals maximum, executes their multiplication and punches the result in an empty area on the same card. It is able to sum and subtract too.



■ **Figure 2.57**
Notice the typical 19th century style in the first model's legs.

1933 A.D.

- **IBM Type 258 Numeric Printing Tabulator**: the first real **IBM tabulator**. High on the left the cards are charged, while the printing device is on the right. In the central part some control switches are visible, and on top five accumulators which the machine uses to sum.



■ **Figure 2.58**
Able to process 150 cards per minute, it printed only numbers. A little management panel is present in the centre of the machine.

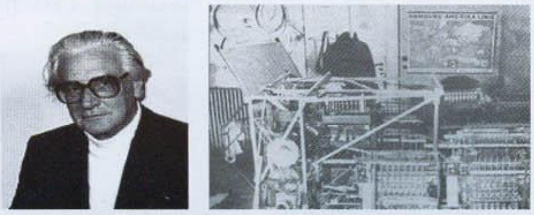
1936 A.D.
- **Alan Mathison Turing**, British mathematician, imagines a "machine" or "**automaton**" (today this appears obvious), existing only in theory, thanks to which he formally demonstrates the possibility of realizing a machine able to solve every algorithm: a calculation or, more generically, a list of operations needed to solve a problem in a finite number of operations. Turing sets up the informatics research that will lead to the Artificial Intelligence. Furthermore, in his written work "Calculating Machines and intelligence" (1950), he proposed a test called "Turing Test", to verify the possibility of the machines to think. Real specimens won't be created, but his idea will be at the base of future computers.

1936 A.D.
- August Dvorak invents **Dvorak's keyboard**, where letters are layed out differently compared to the QWERTY arrangement, due to speeding up the typing (QWERTY system was introduced with the aim of slowing down the typing, to avoid the type bars to jam). In Dvorak's keyboard punctuation marks and vowels were arranged on the left while most common consonants on the right.

■ **Figure 2.59**
Training typists to make them use the new keyboard was too expensive. For this reason, even though this keyboard was much more efficient, it hasn't been used anymore.

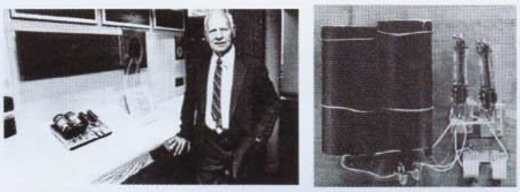
1936 A.D.
- **IBM** sells its first **electric typewriter**. With it later conquers 80% of the global market.

1936 A.D.
- **Konrad Zuse**, from within his bedroom in Germany, starts the building of the V1 logic machine, later renamed **Z1**, to avoid referring to the notorious German V1 rockets. It is the first project of a mechanical calculator. It has been totally fulfilled by Zuse and his family, also in term of costs, with rudimental components. This prototype represents the first completely programmable machine in the world, based on binary code. The machine will become so great in size to fill a lounge too. Zuse is sure that programs composed by bit combinations can be stored. So he asks for a patent in Germany for the execution automatic calculations, including a memory combination. The first real computer was almost about to be born.



■ **Figure 2.60**
Half a second was necessary for a sum to be carried out, and one second for the subtraction, but three seconds were necessary for a multiplication and almost twice as much for a division.

1937 A.D.
- **George R. Stibitz**, American, creates a demonstrative calculator in Bell Laboratories in New York named **K-model**, developing a circuit based on boolean binary logic and using relays.



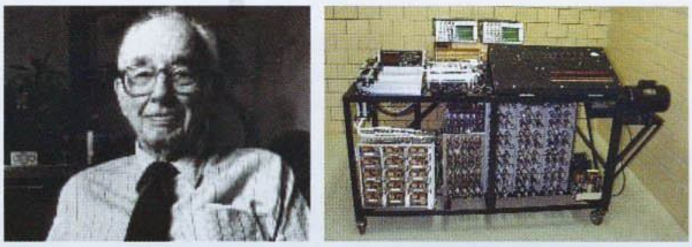
■ **Figure 2.61**
The name, K-model, was chosen by his wife, because it had been built in the kitchen.

1937 A.D.
- **Howard Aiken**, engineering professor in Harvard University, suggests the building of a huge electro-mechanic calculator to Thomas J. Watson, IBM's president. In this period IBM has no interest on computers, since its production is devoted to tabulators, but Watson finances the project for an image reason. IBM starts **Mark I**'s building.

■ **Figure 2.62**
Howard Aiken, US Navy's commander.

1937 A.D.

- **Dr. John Vincent Atanasoff** and his assistant Clifford Berry start building their first electronic digital in the United Sates. It is be achieved in 1942, and named **ABC** (Atanasoff Berry Computer). It is not programmable, but it lays the building blocks for computers to be built in the future.



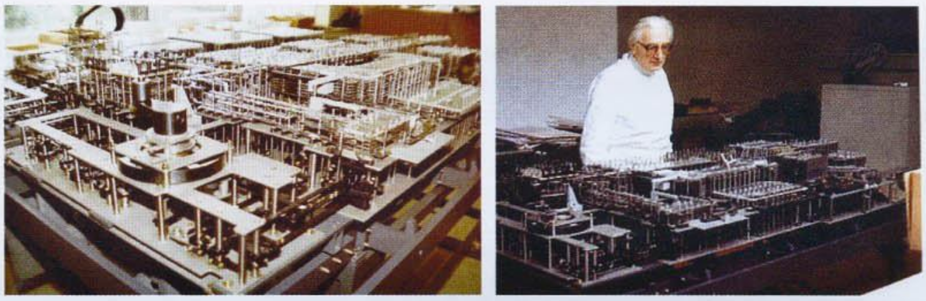
■ **Figure 2.63**
Reproduction of a computer created in 1997.

1938 A.D.

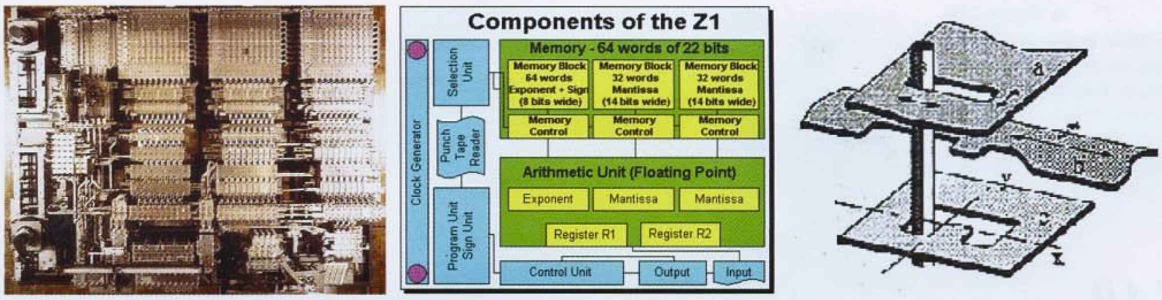
- **Konrad Zuse** completes the **Z1** started in 1936, an electro-mechanic binary computer, and improves the Z2's design. This machine's characteristics are exceptional, considering that each unit is composed by mechanical parts only. It is composed by:

- high efficiency unit for the semi-logarithmic representation of floating-point binary numbers. This would have to be able to carry out calculations of both very little and very large numbers, with good precision;
- high efficiency addition unit, with single step amount to be carried and the management of arithmetic precision;
- memory in which each cell could be addressed by the punched tape and could store arbitrary data. The memory was 64 words long, and each word would contain 22 bits;
- control unit for the supervision of the whole machine, including Input/output devices and from binary to decimal and vice versa passage;

Z1 didn't use relays, but only very thin metallic plates. The only electric part supplied was used to provide a 1 hertz clock.

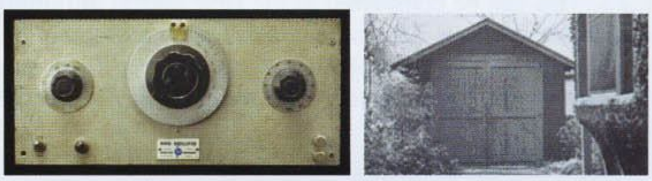


■ **Figure 2.64**
- Z1 detail.
- Konrad Zuse observing his new Z1 model, built with Siemens support, and costing at the time 800,000 Marks (Technology Museum in Berlin, 1989).
- Z1's bird's-eye view. It certainly brings to mind modern chips, with its 30,000 components.
- Z1's block diagram.
- These three plates compose one memory bit.



1939 A.D.

- **William Hewlett** and **David Packard** create the Hewlett-Packard in a garage in Palo Alto, California. Their first product was a sound oscillator was actually built in the garage to be used for Walt Disney's film "Fantasia".



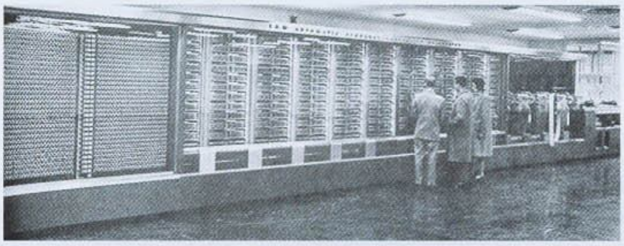
■ **Figure 2.65**
Sound oscilloscope and Hewlett-Packard garage. HP will start to work also on computers only in 1966.

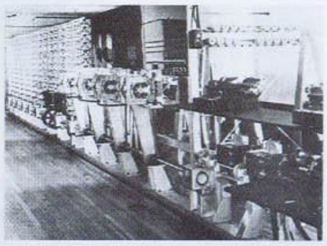
1939 A.D.

- **IBM**'s **ASCC** project (**A**utomatic **S**equence **C**ontrolled **C**alculator) starts. Afterwards it is given up to Harvard University and will be renamed as **Mark1**.

It is 10.60 meters long, 2.60 high, weighing 5 tons, and containing 800,000 parts, with 72 accumulators enclosed and 60 devices with rotating switches, and each one can be used as a constants register. Furthermore a card reader, a punched plate reader and a typewriter. A sum requires 1/3 of second and a multiplication 1 second. The rotating switches are located on the left, followed by the seat of the counting memories. Multiplication and division units and the counters for logarithms and the calculation of trigonometric functions are partially hidden to visitors. The tape unit, typewriter and card punch are located on the right. Several tons of ices per day were needed to chill the machine.

■ **Figure 2.66**
In IBM laboratories, the machine before its shipment to Harvard, which will take place in 1943.

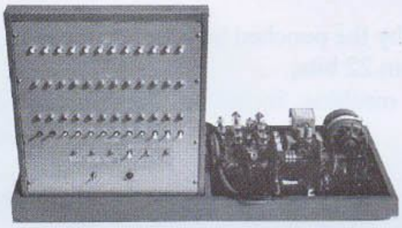




1939 A.D.

- **J. V. Atanasoff** and **C. Berry**'s calculator, **ABC**. **J.W. Mauchly** will base his ENIAC on this calculator. It is the first computer using vacuum valves. This machine, a prototype with 16-bit sums, will never be produced.

■ **Figure 2.67**
Some ABC concepts, such as the ALU and the rewriting memory, will be used in modern computers.



1940 A.D.

- From this year on, because of the **Second World War**, computer evolution changes drastically. Some projects are abandoned or wrecked for this reason, while strategic and military needs boost new studies and machines, like the famous ENIAC, which represents one of the most important results.

1940 A.D.

- **George Stibitz**'team produces several calculators. The first one, named **Complex Computer**, uses 9,000 telephone relays and it is achieved in 1940. It is used for complex numbers multiplications and divisions; it is able to perform a sum or subtraction every 3/10ths of a second. Thanks to a punched tape all the instructions and routines are put in. This system has a sort of time-sharing (processing parallelism).

■ **Figure 2.68**
6 Tickets for data insertion or printing can be connected to the computer. System can manage multiprocessing by linking several computers together to solve different problems at the same time.

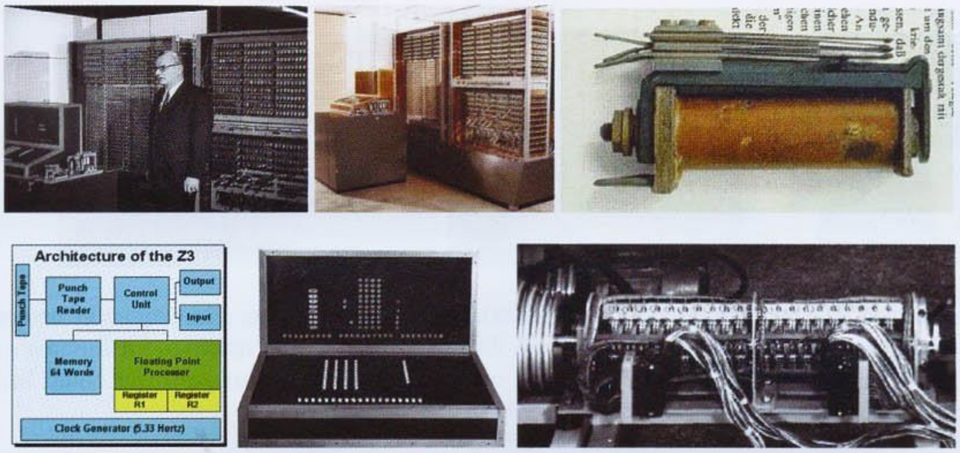
1941 A.D.

- The **NTSC** (**N**ational Television **S**ystem **C**ommittee) outlines standards. These are 525 connection systems.

1941 d.c

- After having completed Z1 and Z2 models, unsatisfied because of their low reliability, **Konrad Zuse** achieves a properly operating model: **Z3**. This can be considered as the **first digital automatic computer**, perfectly working and quite reliable. An important characteristic is Z3 can be controlled by a program based on binary calculus, as imagined by Leibniz. Despite of what has been said about Atanasoff and Aiken, Z3 is surely the first binary computer in the world. Zuse hires a mathematician for programming it: Arnold Fast. Zuse and Fast together will develop Z3 and, subsequentally, Z4 (in 1942).

Arnold Fast is considered the first professional programmer in the world. Konrad Zuse builds Z3 again with him, "Zuse KG Company", between 1960 and 1961, with the aim of demonstrating his machine's real performance, warranting its patent and showing his creature to the whole world. The speed will be variable, but the performance better, because of the presence of relays, which cannot go above 3.5 Hz.



■ **Figure 2.69**
- Konrad Zuse and the Z3, rebuilt in 1961. On the left the memory. On the right the arithmetic unit, with step relays and the console with the punched tape reader.
- The Z3 rebuilt (1961).
- A Z3 relay.
- Z3 design.
- Input and output device. Numbers are inserted by using 4 buttons for the mantissa and 17 for the exponent. Results are displayed thanks to bulbs.
- The Z3 pulse generator (clock).

1943 d.c

- **J.W.Mauchly** and **John Eckert** think that a digital computer could be faster in calculating ballistic tables compared to electro-mechanic ones. Ballistic tables, essential for every kind of cannon and bullet, are necessary for American soldiers. In fact, after the North Africa campaign in 1942, the Allies had understood that artillery shots were very inaccurate, due to the ground's characteristics, which was so different to the American one. At the same time manually calculating all new tables would have been impossible. We can imagine that between 2,000 and 4,000 trajectory calculations were necessary to achieve a ballistic table, and each calculation required roughly 750 multiplications. For this reason ENIAC's contribution was to be so important: it is able to compute a trajectory in 30 seconds only, compared to 20 hours needed by a mathematician using an electro-mechanic calculator. In this way, in April 1943, Mauchly and Eckert, through the Moore School of Engineering in Pennsylvania, show a memo describing an electronic analyzer able to compute trajectories and achieve a table in two days only. The United States Army buys the machine, which will be built in roughly 200,000 man hours.

The machine is named **ENIAC** (**E**lectronic **N**umerical **I**ntegrator and **C**alculator), and will be completed in 1945 and use electronic valves. It does not contain any moving component, except for input/output gears. It has 500,000 welded connections, 18,000 valves, 6,000 switches and 500 terminals. Calculations are performed by generating electric pulses and it uses decimals. The output is given on punched cards. Originally ENIAC didn't have an internal memory, but this possibility was discussed during the building, and finally added. Its numerical word's width is 10 decimals and it'll be able to multiply two numbers having this width at the speed of 300 results per second, finding each result's value in a multiplication table recorded in the memory. ENIAC will be 1,000 times faster than the previous computer generation which worked with relays. The machine is completed in 1945.

1943 A.D.

- **John Von Neumann** deepens a modern computer's needs, by defining its design and developing the first programmable computer with memory. He is sure important advantages and flexibility can be gained just by writing program instructions allowing dynamic changes during the program's execution. This would make the hardware "intelligent". These new instructions, arranged in subroutines, are able to compute much more. The most used routines become reusable, avoiding the programmers to have to entirely rewrite them each time. Programs can be kept undamaged in appropriate "libraries" and loaded on memory when necessary, picking them up from a secondary memory, as punched cards or tapes. Generalized computer's memory becomes a different program's parts assembling area. The first modern electronic and programmable computers using these ideas appear in 1947. These will use the first **RAM** (**R**andom **A**ccess **M**emory), containing 8,192 bytes.

■ **Figure 2.70**
Von Neumann was a talent boy: at six years old he would talk with his father in old greek. At eight years old he knew analysis and at ten he had already read an entire historic encyclopedia.

1944 A.D.

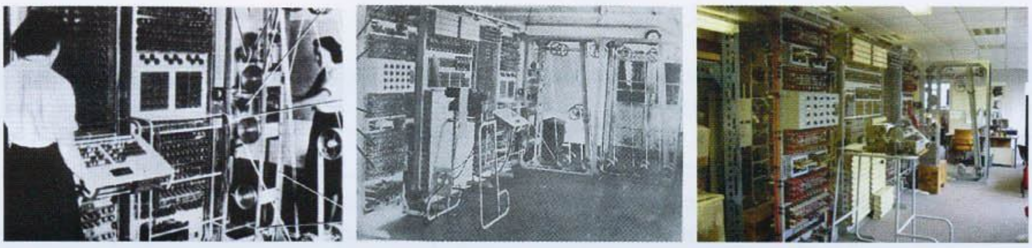
- Germans make use of a codification device called **Enigma**. This works with different keys, which can be set randomly for encrypting messages transmitted by their military commands. This machine had been invented by a polish engineer, and it is not clear how it fell in German hands. English had big problems in deciphering messages tapped out, because of the random keys used by this machine.

■ **Figure 2.71**
German analysts were sure a team of mathematicians would have needed at least one month to decipher one of the 15,576 codes. In the images the machine and near it the rotor.

For this reason Churchill assigned Turing to organize and lead Bletchley Park's cipher comunication center, near London, formed by hundreds of original and brilliant minds. There **Bombe**, a deciphering machine for Enigma's codes, was created. As it did not work very well, M.H.A. Newman, Enigma's code deciphering department chief, hires two British Telecom's engineers, T.H. Flowers and S.W. Bradhurst, to do better.

■ **Figure 2.72**
- Bombe machine, in Bletchley Park.
- Other Bombe image (built again).
- Bombe in detail.

With their help Turing suggests a new calculating method used in **COLOSSUS Mk1** computer, first British electro-mechanic calculator to test at high speed all possible nazi cryptographic machine code combinations. The machine was finished and ready to work on December of the same year. This system appeared to be so fast that history tells us the Allies, during North Africa war, were able to catch messages much before the German command! The name "Colossus" derives from the big quantity of electronic valves used in the machine. The project's complexity and the short time available made for a difficult engineering job. "Colossus" has 1,500 valves and weighs more one ton. It has no memory and it can't be programmed. And yet it is able to manage 5,000 characters per second and decipher each day more than 4,000 secret messages from Germans as well as Japanese and Italians, after having unhinged Enigma's cryptographic system. Churchill does not completely realise the enormous possibilities offered by calculators and Turing's theories. After the war he orders the disassembly and destruction of all "Colossus" models used to defeat nazis. Turing continues studying artificial intelligence and projects new electronic calculators. In the 90's Bletchley Museum's staff rebuilds Colossus again, with the constant opposition of the Secrecy Act, created to protect everything considered as military secrets.



■ **Figure 2.73**
Colossus Mark I prototype. It was assembled in Bletchley Park in February 1944. Colossus Mark II, improved version, was installed in June 1944. Before the and of the war other ten Colossus were built.

1944 A.D.

- Under H.H. Aiken direction, in Harvard University, and together with IBM's technicians, **Mark I** is finished. It is an electro-mechanic calculator, working with relays, entirely automatic and universal. It is considered Babbage's wish fulfilled.

Instructions are given on paper tape, on cards or set by switches. Numbers on which instructions must work are stored on registers. Mark I is an electro-mechanic computer. Its basic operations are carried out by mechanical parts, electrically controlled by 3,000 relays roughly. It starts being used in Harvard in 1944, and keeps working for more than 15 years, producing important mathematic tables, although being outdated for that period.

While Mark I solves mathematic problems for the U.S. Navy, new projects are being carried out to pass from mechanic to electronic data computing.

In the last years a young and very intelligent student, Mrs. Grace Murray Hopper, helped Aiken. She entered the naval reserve in 1943. Thanks to Hopper's help, Aiken achieved the building of Mark I and Mark II, finished in 1947, after the end of the Second World War, and also Mark III. She will leave Aiken and Harvard in 1949 to join Eckert and Mauchly for UNIVAC I's building.

■ **Figure 2.74**
- Mark I.
- Engineers team at work.
- Mark I: complete view of the original model.
- Mark I: left side detail.

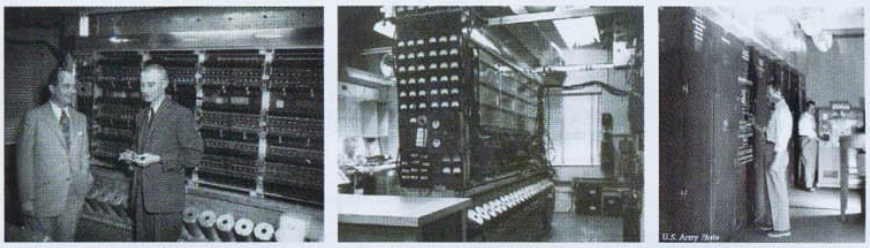
1945 A.D.

- **Grace Murray Hopper**, **COBOL** language inventor. She, or someone among her colleagues, removes a moth, five centimeters long, from Mark II Aiken Relay Calculator's inside, an experimental computer located in Harvard University. It had malfunctioned because of the moth. It is said that the figure of speech "the computer has a bug" when it fails to work properly, derives from this accident.

■ **Figure 2.75**
Grace Murray Hopper, APT language inventor. She will verify, in 1960, the first COBOL language version for UNIVAC.

1945 A.D.

- The **EDVAC** (Electronic Discrete Variable Computer) project starts, under **John von Neumann** and H.H. Goldstine's leadership. It is the first project for an electronic calculator working with memorized program. In other words, it is modern computer's progenitor. It derives from ENIAC, and improves its programmability concept, introducing programs directly inside the memory, instead of inserting them from outside.



■ **Figure 2.76**
- Von Neumann and J. Rober Oppenheimer in front of EDVAC.
- EDVAC.

1945 A.D.

- Probabilistic methods have a long history, but real systematic studies began after 1944 and brought to an excellent development. It has been seen that currently nearly half of high quality scientific applications use methods such as **Monte Carlo** or its recent version, **quasi-Monte Carlo**. This idea, that of systematically using simulations based on probability to solve physics problems, is generally ascribed to Stanilsaw Ulam (1909-1984), Polish mathematician. In truth already **Enrico Fermi**, in 1930, had been this method's inventor, and had used it for neutron's motion calculus, when its name was not yet Monte Carlo. Ulam was an important figure in the American project for the atomic bomb (Manhattan project) during the Second World War between 1943 and 1945 in Los Angeles, New Mexico.

After the war Ulam gave important inputs for the hydrogen bomb. In fact Manhattan project required a huge number of complicated problems to be solved (in his autobiography Ulam describes the moment in which the idea of using random simulations to solve these problems occurred to him while he was playing cards). The algorithm's name, known once as "statical sampling", later became known as Monte Carlo, with reference to the casinò in Monaco. Statistical methods are used by now in many fields, to obtain information and evaluations on phenomena connected to chance. Data comes from computer simulations, able to generate random number sequences. These are used to simulate the uncertain phenomen thousands of times, in order to rapidly collect data that will be statistically treated, giving evaluations which are more and more reliable the higher the number of tests carried out. Quasi-Monte Carlo is an algorithm derived from the previous one. It uses the data in a slightly different way compared to Monte Carlo.

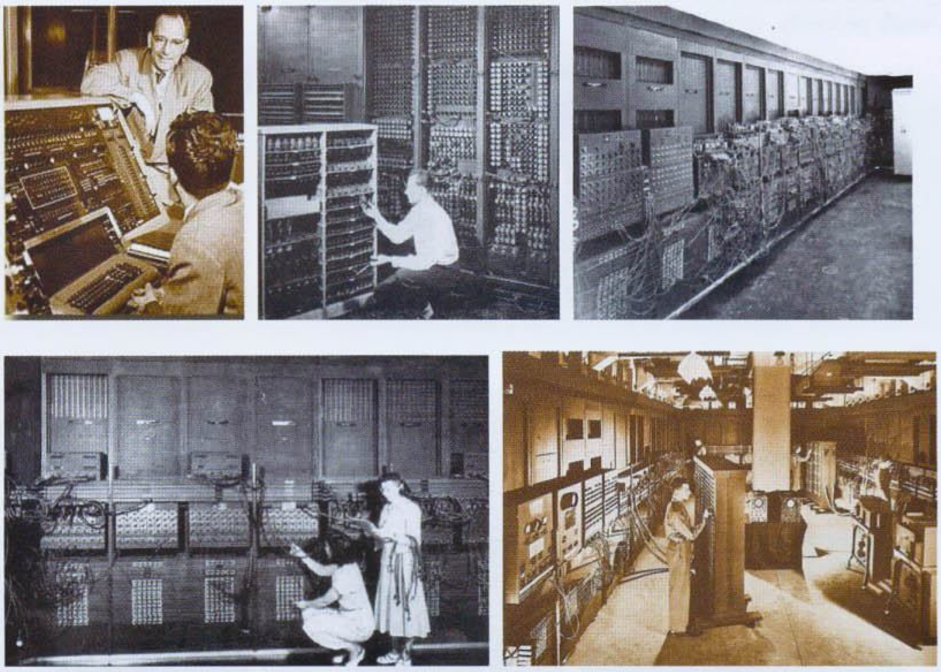
■ **Figure 2.77**
- Enrico Fermi è uno dei padri della fisica nucleare. I suoi studi sulla radioattività indotta gli valsero il premio Nobel nel 1938.
- Stanislaw Ulam.

1946 A.D.

- The use of **ENIAC** begins. Electronic technology enabled to obtain a computing speed one thousand times higher. It is always a machine working with external programming, and is very different compared to modern computers: it is a giant **30 tons heavy**, that breaks very often, because of the vast quantity of frail components.

It is necessary to change cables connections in its insides to program it. Unfortunately we cannot reproduce the smell of these machines here in the book for obvious reasons, deriving from the heat produced, from electric circuits, transformers soaked in insulating oil, cables, wood, overheated metals and several acids. The same goes for the noise. A set of sounds coming from huge conditioning plants, relays, transformer vibrations and circuit power plants.

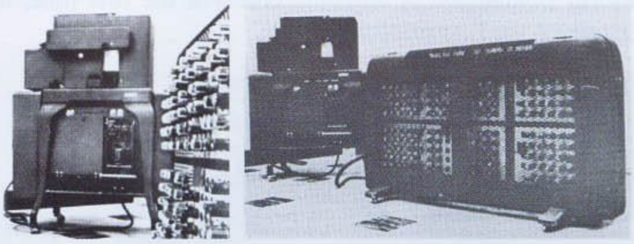
■ **Figure 2.78**
- John Mauchly and ENIAC.
- ENIAC.
- The first electro-mechanic calculator logic is used in the first electronic calculator: ENIAC, which starts in 1942 in the University of Pennsylvania.
- ENIAC and Pentium comparison table.
- ENIAC took up a surface of 160 square meters, weighed 30 tons, and was 10,000 times slower than a modern personal computer.



ENIAC		**150 MHz Pentium**
Speed	5,000 additions/second	300,000,000
Memory	200 bytes	16,000,000
Components	10,000 capacitors 70,000 resistances 1,500 relays	4,000,000 transistors (CPU)
Dimensions Weight	3 meters (height) x 160 m^2 (surface) 30 tons	Personal computer dimensions Some kg

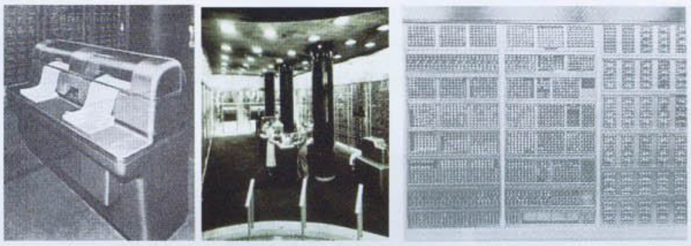
1946 A.D.

- **IBM** develops the **multiplier machine 603**. It is the first mass-produced commercial electronic calculator working with valves. It is able to perform multiplications 1,000 times faster than previous electro mechanic machines. ENIAC's flush is in the air, and for the first time IBM feels outstripped from a project it hadn't been able to anticipate. In answer to ENIAC's success, IBM starts mass-production, where by this we mean about twenty specimens of 603 calculators working at the speed of 6,000 cards/hour, instead of the rival electro mechanic machine's speed of 600 cards/hour.



■ **Figure 2.79**
IBM Multiplier 603.

Anyway, this calculator fails to satisfy Eckert and Thomas Watson Sr., who are in fact marvelled at having not been aware that someone else was producing something without their sponsor. Thus a team is created to project a huge machine: the **SSEC**. Its arithmetic unit is based on 25L6 standard valves used in radios. SSEC's project is to be continued night and day, seven days per week, in Endicott's IBM laboratories. Together with the electronic part, they design a complete group of peripheral devices: high-speed cards readers, tape and cards punches, console, memory units and an enviable management panel.



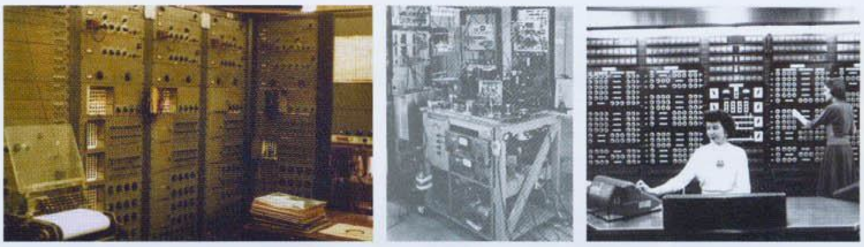
■ **Figure 2.80**
- SSEC's printing machine.
- Thomas Watson Sr. believed the three central columns could ruin the aesthetic quality, thus he ordered to touch up the advertising pictures, removing them.
- SSEC's mangement panel.

1947 A.D.

- **ENIAC** is installed in Aberdeen's balistic research laboratory. It is the first **general-purpose**, that is to say universal, digital calculator. It is programmable from outside on wide scale. Since it is **1,000 times faster than Mark I**, it is used for weather predictions, planning, balistic tables, etc…

1947 A.D.

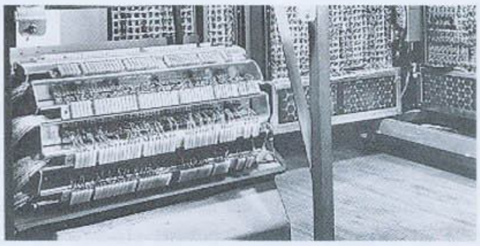
- In July, **Howard Aiken** and his team complete **Mark II** in Harvard.



■ **Figure 2.81**
Several Mark II pictures.

1947 A.D.

- The **magnetic memory drum** is introduced as a device fit to store data in the computer.



■ **Figure 2.82**
Magnetic memory drum.

1947 A.D.

- In 1944 U.S.A. navy finances a **Massachusetts Insitute of Technology**'s project aimed at creating a completely electronic **flight Simulator**.

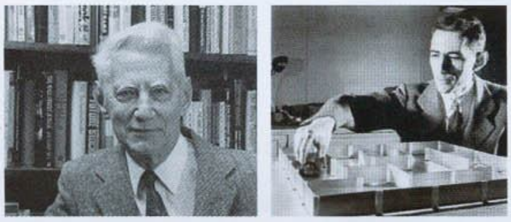
The result of these efforts was the **Whirlwind**, first computer able to react in real-time to user's actions, instead of awaiting inputs and giving answers. **Jay Forrester**, project leader, had guessed the major importance of having a real-time computer compared to having a flight simulator.
In 1947 Forrester asks support from the Navy to make this project larger, aiming to create a general purpose computer. He obtains one million dollars per year, and he demonstrates the final version of Whirlwind in 1951: eight valve closets. The performances are excellent, comparable to the 80's personal computer ones, for example the TRS-80. It is also to be the first computer to be used as a personal computer: it is then possible to put one's name down for using it for 15 minutes to run a program, perform simulations or anything else. For the first time, it will use ten **magnetic cores**. It starts working in 1950. It is considered the first **minicalculator**.

■ Figure 2.83
The first computer working in real-time, born as a flight simulator, but then become the first United States radar defense computer system "brain": SAGE (Semi- Automatic Grounding Environment). Whirlwind was also the first computer having an interactive video interface.

1948 A.D.

- **Claude Elwood Shannon** issues his most important work, **Comunication's Mathematic Theory**, one of the pillars of modern information (and also computing) theory. Once again there is a practical problem. How to transmit messages avoiding noise to ruin their content? First of all, the problem was defining what the message's information content was. Shannon's astute idea was to identify the contents of the information no longer with the contents of the message, but with **the number of 0s and 1s** necessary to transmit the message. The nature of the message is no longer important: it could be numbers, music, words. In any case it is a sequence of 0s and 1s. The term **bit** compares in this article for the first time.

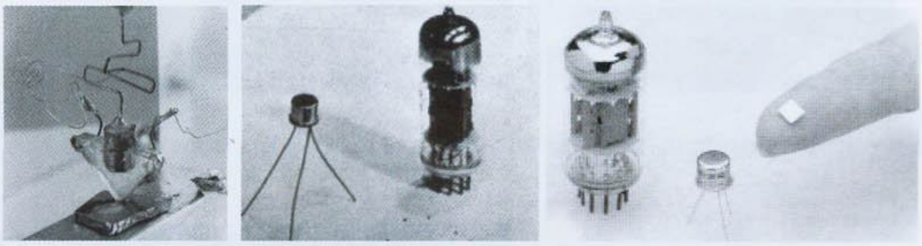
■ Figure 2.84
Shannon's electro-mechanic mouse (Theseus) has been one of the first attempts to "teach" a machine to learn. It is one of the first practical experiments on Artificial Intelligence.



1948 A.D.

- **Walter Houser Brattain**, **John Bardeen** and **William Bradford Shockley**, American physicists, achieve the **transistor** (Transfer Resistance) in Bell Laboratories. In 1956 they will receive the Nobel Prize in physics for this job. The transistor is the first electronic device paving the way to electronic miniaturization, and, as a consequence, to digital and informatic revolution too, by replacing valves. But many other researches still have to be carried out before production of transistors: six years will pass. Small size, high reliability, low heating and low production costs guarantee the transistor's success. Transistors are to make computers 1,000 times faster in comparison with the ones of the preceding epoch.

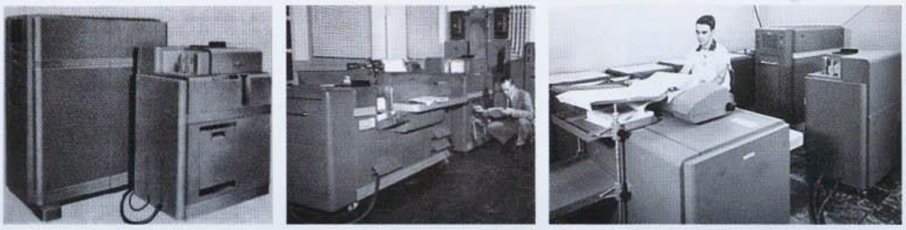
■ Figure 2.85
- First transistor.
- A transistor and a valve.
- Comparison between an electric pipe, a transistor and a modern chip, able to contain millions of transistors.



1948 A.D.

- The **IBM 604** electronic calculator is conceived for commercial calculations. In the following ten years 5,600 units will be sold. It is described as a "miniaturized" calculator, but in truth it takes up 2 x 2 x 1 meters and weighs 640 kg. In its basic configuration it is used with a reader card punch. To make it useful for commercial purposes it is linked together to a 402 or 407 tabulating machine and from one to three 941-type memory expansion units. All this set is to be easily sold with the name CPC (**C**ard **P**rogrammed electronic **C**alculator). The price is around one million of current dollars. 604 has a clock speed of 50,000 pulses/second (today 50 KHz). 402-407 and 941 machines are electro-mechanic, and use relays and rotating counters with times of 400ms per operation, versus 604's 0.5ms. Machine's downtimes, as well as the preventive maintenance, goes from 10 to 15%. Most of the 1,400 valves of which it is composed are double triodes: these cause most of the problems. This machine will be mass-produced in the 50's in Amsterdam's IBM plant.

In the system's full version, the CPC, problems of some complexity can be solved too, as for example finding the squared root of a polynomial with seven complexity degrees.



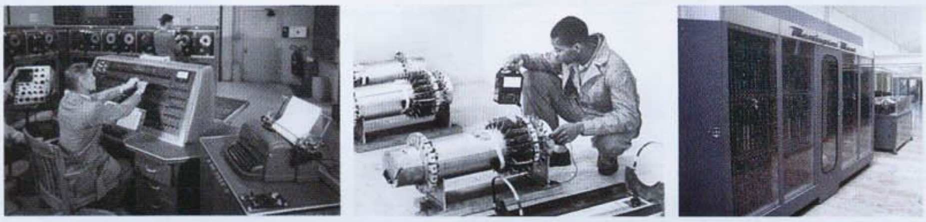
■ **Figure 2.86**
- IBM 604 calculation unit and card reader/punch.
- IBM CPC 604.
- IBM CPC, composed by 605-type units (calculation), 417 (tabulating machine) and 914 (box containing 480 digits of additional memory).

1949 A.D.

- **Sony** is born in Tokyo, thanks to **Masaru Ibuka**, engineer, and **Akio Moria**, physicist. The initial funds were of 190,000 yen used to create a company with 20 employees, instructed to fix electric devices and try to manufacture their own products. The final turning point towards success comes about in 1954, when Tokyo Tsushin Kogyo K.K., or Tokyo Telecommunications Engineering Corporation (the company is later renamed this way) obtains a licence for transistor production.

1949 A.D.

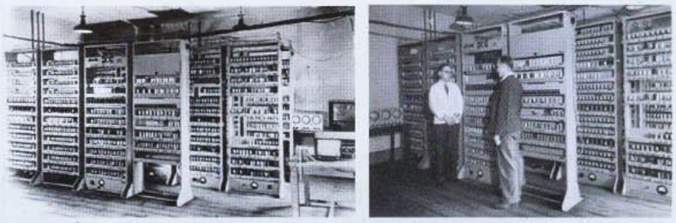
- **Eckert** and **Mauchly** found the **UNIVAC-1** company, to develop the UNIVAC-1 (**UNIV**ersal **A**utomatic **C**omputer), based on the idea of a computer with a memorized program. It is the first company determined to produce computers on a large scale no more for military or scientific aims only. 46 units are produced and sold for more than one million dollars. The size is smaller in comparison to Colossus and ENIAC; they are in fact the size of a large closet, and no more of a large room size. The room is in any case necessary to contain the console, the printing machine and the storing units on magnetic tape. From the birth of this machine onwards, programs set with cables or switches will become outdated. UNIVAC is a decimal system computer. Each memory has 100 words, and in total 1,000 words, each one 12 decimals long. 7 binary bits are present for each cipher, parity bit included, 2 bits for alphabetic codification, and 4 bits for a decimal number representation.
The machine receives the instructions directly from the program stored in itself. The "Short Code" will be both the first interpreted program and the first Assembly-type program.



■ **Figure 2.87**
- UNIVAC-1.
- A technician testing a memory unit in UNIVAC-1.
- UNIVAC specimen present in the Monaco museum.

1949 A.D.

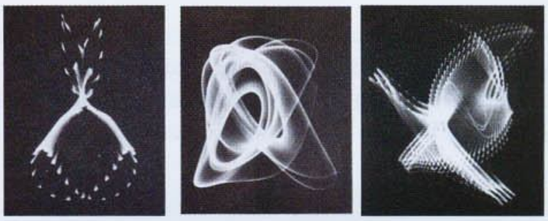
- The **first electronic automatic calculator with a memorized program**, the **EDSAC**, is built in Cambridge: it contains 3,800 valves, and a memory of 16 words 35 bits long. The machine corresponds to the UNIVAC. It contains acoustic memory pipes, an oscilloscope as display and the subroutines library designed by Wilkes. The library consists of short programs called subroutines and probably represents the first system's kernel effort. It can be considered as the first calculator working with a totally memorized program.



■ **Figure 2.88**
EDSAC's programs and data can be changed in memory, exactly as Zuse's patent suggested in 1936, but Zuse himself did not perform this technology in his Z1, Z2 or Z3.

1950 A.D.

- **Ben Laposky** uses the oscilloscope for art, using lines and waves after having it photographed.



■ **Figure 2.89**
Primordial graphic art.

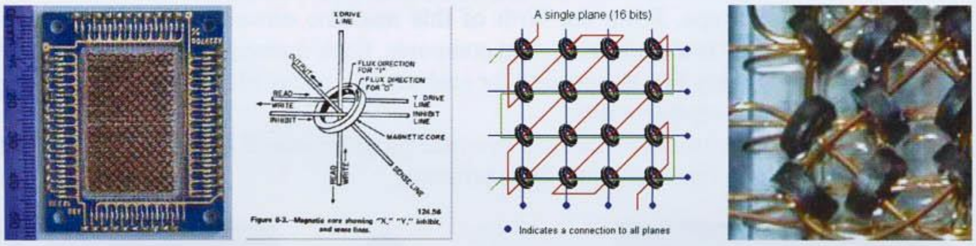
1950 A.D.

- **Mark III** project is achieved. It is based on electronic valves only. Computer picks up data from a magnetic tape. The machine weighs 35 tons and is composed by 700,000 single parts.

■ **Figure 2.90**
An addition requires 0.3 seconds whilst a multiplication 6 seconds.

1951 A.D.

- **Jay Wright Forrester**, the **Whirlwind** project leader, patents the **magnetic nuclei memory** with the name of "Multi-coordinate Digital Information Storage Device" on the 11th of May. These memories are composed by little magnetic nuclei through which 4 threads pass. The blue crossed threads change nuclei's polarization. Electricity passing in one thread only wouldn't be enough to invert the nuclei's polarization, so two threads are necessary, and when electricity in both threads meets in the crossing point, it is strong enough to change direction of the magnetic field, thus passing from 0-state to 1-state or vice versa. The memory reading system is destructive, because it changes nucleus polarization, which has to return in its original condition. In any case it is a amazing invention, because memories are not volatile (being formed by magnetic fields), very rapid and certainly reliable. The construction of small nuclei was carried out manually, and this was no simple job, considering the size and the big number of them built in the following years.

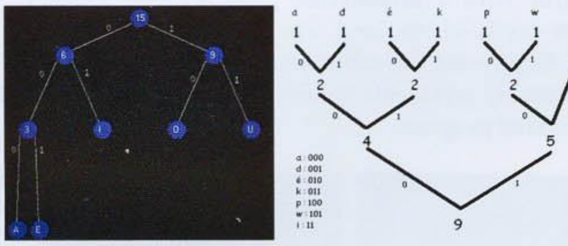


■ **Figure 2.91**
- A nuclei memory plan, of 3 squared cm.
- A memory nucleus functioning scheme.
- A single plan formed by 16 bits.
- Detail showing threads passing through nuclei.

1951 A.D.

- **David A. Huffman** (1929-1999, USA) develops **Huffman's Code**.

This method will be used to press data to be sent on web via modem, in videotape recorders and high definition TV programming. This algorithm allows the boosting of data up to an extra 25 %. This is useful to economize time and money. Easily understandable, considering transmissions were carried out at 110 bits per second. The **binary tree**: simple idea. It is enough to assign the shortest binary code to the most used characters. This process is achieved thanks to a sort of binary tree.



■ **Figure 2.92**
Huffman's binary tree example.

1951 A.D.

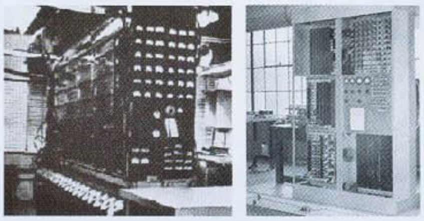
- After few years of development, in England the first commercial generic computer starts working: **LEO**, a commercial version of Cambridge's **EDSAC**, built by **Lyons Company**, an English catering company involved in commercial computer construction since 1947. Its power consumption was 30,000 watts. The machine is provided by 228 pre-assembled units, connected in 21 racks. This arrangement made the access to the components easier in case of damage. This was in fact the biggest problem: the machine was unreliable, because of the valves. Nearly 50 valves per week had to be changed!

The machine also had a loudspeaker so that the programmers could hear all LEO's sounds while he worked. Programmers then became such experts in distinguishing some frequency variations that they came to be able to understand when something is malfunctioning just by the sound. Thus programmers created something we could call "**computer music**". **LEO II** and **LEO III** will be produced later on.

■ **Figure 2.93**
- Leo I.
- Leo II.
- Leo III.

1952 A.D.

- Mauchly, Eckert and Von Neumann complete **EDVAC** in Moore School. They run its **first production program**.



■ **Figure 2.94**
EDVAC.

1952 A.D.

- **Thomas Watson Jr.** becomes the new **IBM president**. Starting from this year IBM decides to add computers to its commercial production line. This enables IBM to become a main element in this sector. It is also charged for monopoly in the area of commercial computers, in violation of the Sherman Act.

■ **Figure 2.95**
The first computer production line will concern the 701 model.

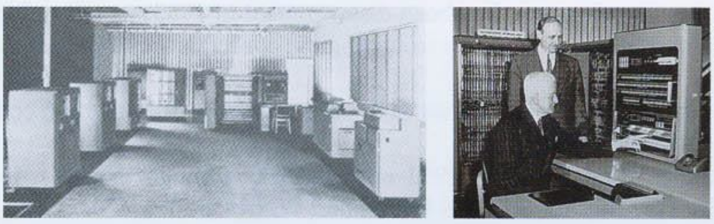
1952 A.D.

- **Mr. Potato** was invented. He later becomes one of the main characters in the Pixar: Toy Story.

■ **Figure 2.96**
In the first selling year this character, created by George Lerner, made the company earn 4 millions of dollars.

1952 A.D.

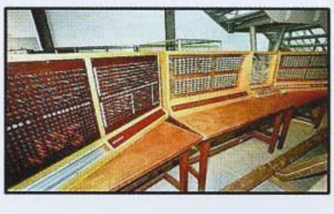
IBM announces **701**, a new calculator conceived for scientific calculus, whose first specimen is to be used for USA defense. In its lifespan, lasting 3 years, only 19 specimens will be installed.



■ **Figure 2.97**
Some 701's pictures.

1952 A.D.

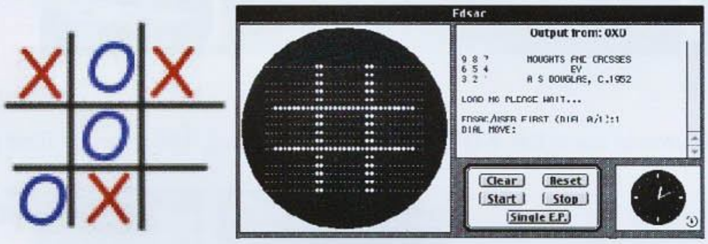
- **W.L. van der Poel** built the first calculator in Holland. Building started in 1947 and was achieved in 1952 thanks to TU Delft students, who worked on Van der Poel's designs.



■ **Figure 2.98**
The machine was based on relays, and its size was 5 m x 60 cm. It took 30 seconds to carry out a sum, and 45 for a multiplication.

1952 A.D.

- **A.S. Douglas** wins a master in Cambridge University. In the university an EDSAC is present, so Douglas decides to write his thesis on Human-Computer interaction. He performs a graphic game, **Tic-Tac-Toe**, Tris, which was visualized on the computer's screen. This could be the first computer graphic game known. Two people can play, or one can play against the machine, which uses special algorithm to win when possible.



■ **Figure 2.99**
At that time the game could only be run on the university EDSAC computer.

1953 A.D.

- **Peter Goldmark**, from CBS, invents **color television**, using a system based on the three basic color filters: Red, Yellow and Blue. CBS starts its color transmissions.

1953 A.D.

- The biggest computer working with valves ever built was for the USA aviation project named **SAGE** (**S**emi-**A**utomatic **G**round **E**nvironment), for the defense of american territories. Started in 1953, it was completed and able to work only ten years after.

In the 50's MITRE's founders played an important role in SAGE system development. It was created as a new air defense system for United States protection from long range missiles or other weapons. SAGE system collects information from radars stationed in different points in the American territory, and then sends them to the central station via telephone line, to be elaborated. As the system grows, new radars, communications, computers, visualizes and new program technologies are used. How many programmers exist in the world in this epoch? Well, it seems 20% of all the programmers are working on the same project! From 1953 and for ten years after, SAGE is to be the most challenging project amongst them all, and it employs 800 programmers and technical resources of many big American companies, IBM above all. It is hard to believe, but very few people had knowledge of this project in those years. Air control necessity arose from the cold war, and its realization was assigned to two MIT professors, Jay Forrester and George Valley, in Lincoln Labs. After having jumped the gun several times and after several project changes, the system that went in was composed by 23 bunkers in reinforced concrete, spread on the American territory, and one in Canada, all linked to a collecting center for the air defense. SAGE's task was tapping missiles transporting Sovietic atomic bombs, and guiding American aviation to destroy invaders. It had to be linked to Bomarc and Nike missiles bases too.

Each Direction Centers bunker contained a computer, called "A/N FSQ-7". It used 3 megawatts of power, it had 60,000 valves roughly, and it employed 100 operating people.

It has been estimated this project has cost between 8 and 12 billions of 1964 dollars, much more compared to Manhattan project, which developed the first American atomic arms.

The most important suppliers responsible for SAGE were: IBM for the harware, Burroughs for communications among different centers carried out via modem, MIT's Lincoln Lab, later renamed MITRE Corp., for systems integration, the Western Electric for project and construction of buildings, and the SDC, a Rand Corp. Society, for the sofware.

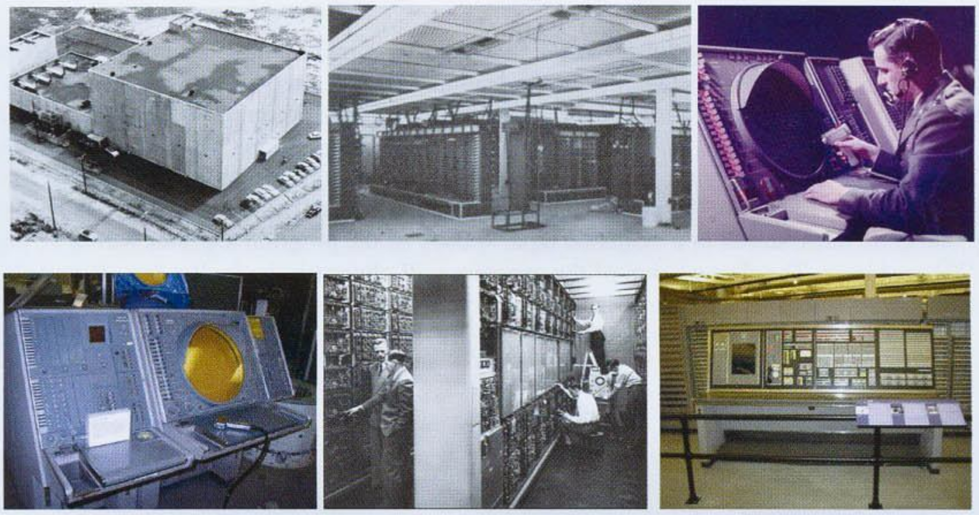
The first SAGE Direction Center started working in November 1956, and the last one was ready by the end of 1962. When the last SAGE center, the Canadian one, had been disassembled in 1983, its technology, historically recognized, appeared to be extremely important because of all the technological consequences that derived from it, among which:

- the invention of the modem and the way it has been used to connect all direction centers sharing data related to different air sectors conditions;
- magnetic memories, then used, between 1953 until halfway through the 70's, in every computer for high speed, invented by Jay Forrester for SAGE's forerunner: Whirlwind I;
- the longest program ever written: a real-time procedure, unimaginable for that day and age, 500,000 lines long;
- Its reliability, considering that mainly other computers had a maintenance downtime lasting one month; instead SAGE had been offline from 1 to 10 hours per year!
- wide development of A/D and D/A conversion techniques;
- the COMPOOL, a system memory area available for several subroutines. This had been invented specifically for SAGE. COMPOOL will be a basic concept for COBOL;
- the use of functions, which we call "multiprocessing", "database management", "distributed processing", "timesharing", "interactive display", and "networking";

- Preventive maintenance: a preventive diagnostic control system to guess future damage;
- Bufferized I/O;
- "cycle-stealing" memory.

"It has already been done for SAGE", this has been a frequent sentence in all industrial environments for many years. SAGE represented a 500 millions dollars affair for IBM too, beyond having formed an entire generation of technicians. It is believed that without this profit the IBM wouldn't have had all the money (and know-how too) necessary to invest in developing the **"System/360"** system series, which is to become the most successful computer architecture through the years and thanks to which IBM will earn more than 100 billions of dollars! SABRE's aerial booking system derived from SAGE and obviously SAGE's technology was used in American civil air traffic control. System's characteristics are:

Architecture: duplex CPU, no interrupts, 4 index registers, Real Time Clock;
Word Length: 32 bits;
Memory: magnetic core (4 x 64K words); Magnetic Drum (150K word); 4 IBM Model729 Magnetic Tape Drivers (roughly 100 words ea.); all systems use parity checking;
Memory Cicle Time: 6us;
I/O: CRT display, keyboard, light gun, real-time serial data (teletype, 1,300 bps modem, voice line);
Technology: electronic valves (60,000), diodes (175,000), transistors (13,000);
Dimensions: CPU (50 x 150 feet, each), consoles area (25 x 50 feet);
Weight: 250 tons (500,000 lbs).



■ Figure 2.100
- SAGE, acronym meaning Semi Automatic Ground Environment, is a detecting and intercepting automatic system of NORAD, used against enemy's planes between the 50's and the 80's. Once the system was completely ready to work, the threat of Soviet planes had already been replaced by the international missiles threat.

- In the images, one of the bunkers in c.a., the SAGE, an operator in front of a computer, a station computer, operators at work and SAGE's control console.

1953 A.D.

- **IBM 650** comes onto the scene. It is known also as a **magnetic drum calculator**, and it becomes the **first industrially produced computer.** We can say that the **first minicomputer** is born with this machine. 450 specimens will be sold in the first production year. More than 1,500 will be sold in the following 15 years, a real record. As the 701 model could do, the 650 is able to write and read both from magnetic tape and punched cards.
Punched cards are also used to insert the program to be executed, and each card represents an instruction with three addresses, that is to say it contains the address of the following instruction too. In 1956 its rental price was 3,200 $/month (the price of a big Cadillac). The system's uptime was 80% (not warranted). It was able to execute a sum or subtraction in 1.63 ms, a multiplication in 12.96 ms, and a division in 16.90 ms. In the most of the systems the memory was the magnetic drum, containing 2,000 words (10 digit + sign), and a random access time of 2,496 ms. Although IBM 650 wasn't a great machine, it had a particular characteristic that made it appealing and easy to sell: lots of flashing lights. Thanks to these lights anyone could realise something was moving and happening in its core. Many people ascribed the success of the machine to these lights and to the fact the machine used 80 columns of punched cards, compatible with all other IBM machines, amongst which tabulators, necessary to print computing results on paper.

■ **Figure 2.101**
- The IBM 650's magnetic drum memory.
- The IBM 650's console and lights, an interesting way of reading results.
- The IBM 650's calculator.

1954 A.D.
- In January a first attempt of translation from Russian to English was done by using an **IBM 701** system. Results were backed by the employment of an electronic dictionary and by several programs containing syntax and rules of the two languages.

■ **Figure 2.102**
A good translation was obtained and much of the experience gained in this attempt will be used in Echelon's project.

1954 A.D.
- In Italy **RAI** starts regular TV transmissions.

1954 A.D.
- **John W. Backus**, an IBM project-chief, creates **FORTRAN**, a scientific programming language (**FOR**mula **TRAN**slation) for the IBM 704 system. Backus and his team feel quite prepared enough on their previous researches, so as to issue a paper titled "Preliminary Report, Specification for the IBM Mathematical FORmula TRANslating System, FORTRAN". Together with other IBM technicians, Backus introduces this new programming language to customers who had already bought the 704, with the aim of obtaining criticism and suggestions on its functions too. Thanks to this idea Backus managed to anticipate the completion of his compilator by 6 months, instead of taking 2 years, as prefixed. The compilator consists of 25,000 code lines, stored in a magnetic tape. A copy of the program together with a 51 page-long guide are given to each 704 customer. The first version of the program will obviously be faulty, but afterwards all the bugs will be removed. In this way scientists can work independently from programmers, by inserting programs directly into the computer. It will be necessary to wait until 1956 to see the first real FORTRAN guide appear. Documenting is the last thing programmers think about.

■ **Figure 2.103**
Scientists and engineers will definitely choose this way of programming, abandoning the method of modifying cables to change programs.

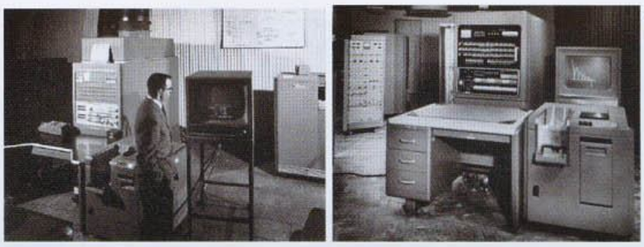
1955 A.D.
- The first **optical pen** is added to system **SAGE**.

■ **Figure 2.104**
The first optical pen in history.

1955 A.D.
- The first **optical fibre** is achieved. Each thread has the size of a hair, and is composed by two different types of glassy materials: a transparent internal one, and a reflecting external one.

1955 A.D.

- *IBM* announces its **704**, a new calculator with nuclei memory instead of the common CRT used in the previous IBM 701 systems. It contains floating-point arithmetic, and many new instructions for scientific calculus.



■ **Figure 2.105**
IBM 704.

1955 A.D.

- **IBM 702**: the first commercial machine completely built with transistors and put on the market by IBM. The line of development has now been traced. Many "first" machines using only transistors will appear, but this is definitely the first one.

■ **Figure 2.106**
The high costs of transistors will be the cause of their flop.

1955 A.D.

- William Henry Gates III is born in Seattle. He is well-known as **Bill Gates**.

■ **Figure 2.107**
Together with his friend, Paul Allen, he created a first company in 1972, named Traf-O-Data. Its task was monitoring Seattle's urban traffic via computer. Always with Allen, Gates develops a version of the language BASIC for the first personal computer: the Altair 8008.

1955 A.D.

- **Steve Jobs**, future founder of Apple Computer Inc., is born.

■ **Figure 2.108**
Founder of Apple Computer and Pixar, he is to become famous for having introduced the first personal computer with mouse and icons interface to the public.

1956 A.D.

- **IBM** introduces and begins the installation of **RAMAC 305** (**R**andom **A**ccess **M**ethod **O**f **A**ccounting **A**nd **C**ontrol). This will be the first step towards a new way of transferring data, called records: from old and cumbersome punched cards support to magnetic disk units. This is a revolutionary passage, because it permits the operation of adding, updating or erasing of records just by rewriting them on the disk. Besides being cumbersome and heavy, cards didn't enable a direct access to data, were bound to 80 columns, could not be changed and, last but not least, were expensive.
In any case, RAMAC 305 is not the machine able to prompt the passage from cards computing to magnetic tapes and/or disk systems. This machine is very expensive and quite frail and therefore maintenance downtimes often occur. It is also difficult to program it, because of the necessity, for setting a job, both of loading a program still on cards and setting several removable wire jumpers. These are the first commercial computers equipped with a unit of **fixed disks** for the storage of data instead of magnetic drums or tape units. The system widely uses electronic valves, positioned in baskets which are easily removable. The disk unit is made up of a stack of **50 24"-discs**, with total capacity of 5 or 10 millions of characters (5Mb or 10Mb), which is a great deal for that period. Rotating speed is 1,200 rounds/minute. The unit could have one or two access branches; the second one was optional. Each branch has just one reading/writing head.

During an operation of data research on disks, the mechanical arm, commanded by pressurized air, has to move firstly vertically to reach one of the 50 disks, and then horizontally to reach the desired track. Data writing or reading is carried out at the speed of 22,500 char/sec. The times of access vary from 100 to 800 milliseconds. In case a head brakes several days are necessary for repairing it. In any case the whole device gets taken to maintenance once a week. The original name is fixed disk; the term "hard disk" will arise around 1970 in opposition to the newborn floppy disks.

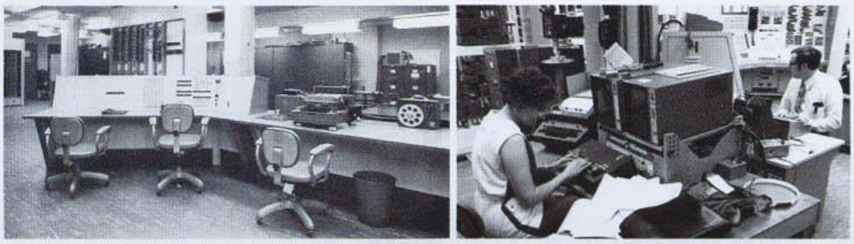
■ **Figure 2.109**
- Printing machine, card reader, computing unit, disk unit, console and typewriter.
- A complete workstation. A HD was high 50 cm!
- An HD detail.

1956 A.D.

- The first computers working with transistors are announced: it is the **TRIDAC** and three experimental models with the initials **TX-0**. A **transistor-UNIVAC** is introduced too, specifically for commercial purposes. These new computers based on transistors pave the way to the **SECOND GENERATION OF COMPUTERS**.

The **TX-0**, developed by Lincoln Laboratories in Massachusetts Institute of Technology, widely uses transistors. It is in fact the first wide-scale experiment of a computer based on transistors with such a system memory. The system is a transistor version of Whirlwind, another Lincoln Labs project. But while Whirlwind would fill a whole building floor, the TX-0 takes up a single room, and is fast. Like the Whirlwind, the TX-0 is equipped with a screen positioned in an oscilloscope container of 12 inches. The output is a 512 x 512 pixels screen inside a 7 x 7 inch screen.

Once the TX-0 was finished, the attention of the laboratory was focused on the **TX-1** project. This project will have to face many problems, due to its complexity, and will then be turned into an easier one, the **TX-2** in 1958. The project of the TX-2 will continue, contending with many problems. One of these is due to the fact that many members of the team will abandon the project to found their own companies. Shortly after, the modules of the TX-2 are individually sold to the Digital Equipment Corporation (DEC), which will decide, in 1961, to produce a cleaned-up version of TX-0 called PDP-1. The first PDP-1 is to be installed in a room near to the TX-0, and they will work together for several years.



■ **Figure 2.110**
The TX-0 and the TX-2.

1956 A.D.

- Thanks to **Alex Poniatoff**, Ampex develops the VR1000, the first **video-tape recorder**, which will enable broadcast recordings. The way of conceiving programming changes and TV nears what it is currently. Ampex recorder will be sold since 1956 earning 50,000 dollars. 15 years later the first household video-tape recorders appear, and the fight between Vhs and Betamax explodes.

■ **Figure 2.111**
- Ampex VR1000 working in the CBS studios.
- Ampex VR1000 and its creator, Alex Poniatoff.

1956 A.D.

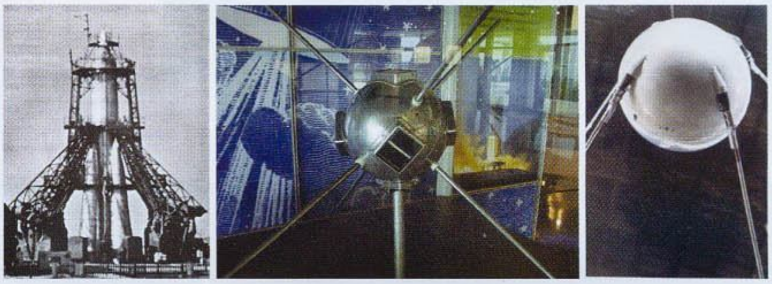
- The **Secam** system is developed in France. Instead in Italy the **Pal** is to be chosen, becoming the most common one. The delay in the employment of colors in Italy (1976/77), while color TVs were already being sold in 1970, will contribute to the failure of the Italian electronic industry.

1957 A.D.

- **FORTRAN-1** is formally issued. This product, the first one among high-level languages, was developed by John Backus and his team in IBM. FORTRAN uses a notation very similar to that of algebra. For this reason FORTRAN will become so popular among scientists and technicians.

1957 A.D.

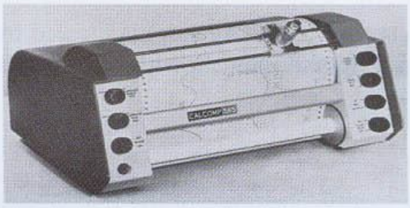
- Russia launches the first satellite into orbit, **Sputnik I**, and the rat race for space-travel starts.



■ **Figure 2.112**
The Soviet Union overtook the United States with the Sputnik I: in fact the United States managed to send their first satellite into orbit, Explorer 1, only later, on the 1st November 1958. The instruments present on Sputnik I continued working for 21 days.

1958 A.D.

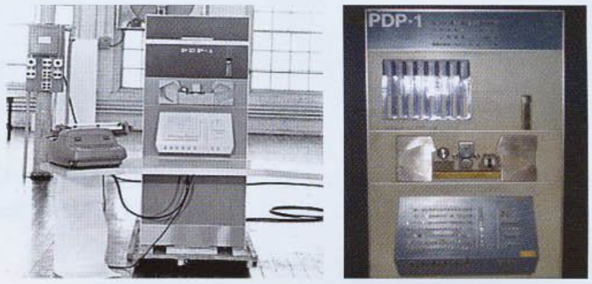
- A little plotter, **CalComp 565**, is produced. Each passage caused the shifting of the nib of 0,1 mm in positive or negative direction up to 250 steps for line.



■ **Figure 2.113**
The plotter was usually sold together with IBM 1627.

1958 A.D.

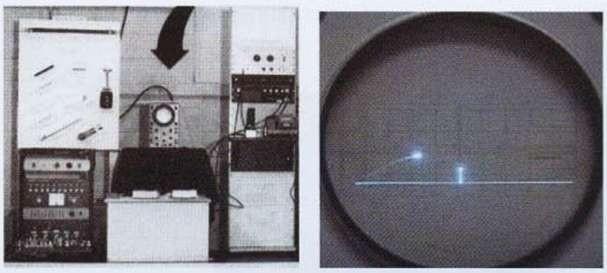
- Ken Olsen and Harlan Anderson found the Digital Equipment Corp. Their first computer, the **PDP-1** (Programmed Data Processor) will be released in 1960 and it will have many successful progenies. The DEC is to be one of the most important computers produced worldwide. And, most of all, the standard of PDP-1 is to be open: all constitutional details are to be available, in order to supply "advanced" users with the possibility of personalizing or improving the machine when needed. Incidentally, this will always happen.



■ **Figure 2.114**
At the beginning, the market was distrustful: the first programmable computer on the market sold just 49 specimens. But technically, it was a success: it was provided with an integrated cathode tube screen, it was possible to put it in a small room, and it managed to supply a decent calculation power, considering its cost: "only" 120,000 dollars.

1958 A.D.

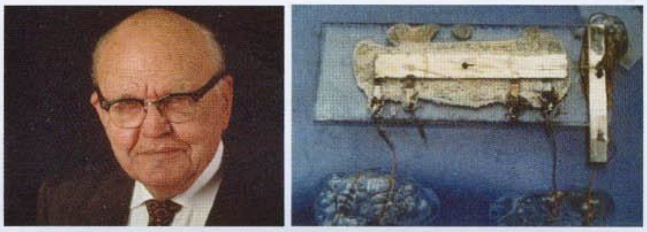
- William Higinbotham achieves, in the Brookhaven National Laboratory in Upton (New York, USA), Tennis for two, the **first videogame** based on a computer and a screen. His aim was to improve relationships with citizens in his area, who were suspicious about the nuclear research activities carried out in Brookhaven. An oscilloscope generates the image and a computer working with analogue valves computes the trajectory of the "ball", crossing an extremely personalized tennis field (the "net" is simply an overturned T). But nobody is particularly interested in a commercial version of the game and Higginbotham does not patent it. Many informatics computer historians retain it can not be considered the first videogame, since it had not been projected for household use or any other form of marketing.



■ **Figure 2.115**
The game was exposed for laboratory visitors for two years, arousing so much interest that queues of people wanting to play spread everywhere inside of the building.

1958 A.D.

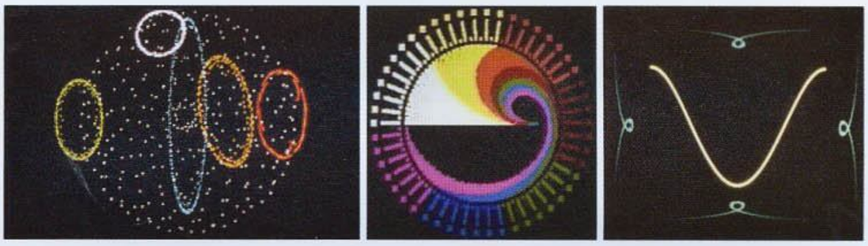
- ***Jack St. Clair Kilby*** from Texas Instruments and ***Robert Noyce*** from Fairchild Electronics invent the integrated circuit, IC.



■ **Figure 2.116**
Robert Noyce and the first IC specimen in the world.

1958 A.D.

- **John Whitney Sr.** uses an analogue computer to create **artistic images**.



■ **Figure 2.117**
He will produce his masterpiece, Arabesque, in 1975, thanks to a computer belonging to a new generation.

1958 A.D.

- The **ARPA** (**A**dvanced **R**esearch **P**rojects **A**gency) is created by the defence department of the USA. Its birth is the American answer to the Russian Sputnik launch, with the aim of exploring computing possibilities applied to military purposes.

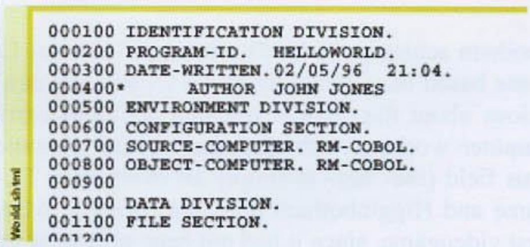
1959 A.D.

- The **IBM 709**, the last big scientific computer of the first generation, still working with valves, is produced. It is the first machine equipped with a **data channel** for I/O. The central unit can be opened like a book, making the access to cables easier.

■ **Figure 2.118**
The complete system is composed by seven units including memory, data channel, power pack, card reader, printing machine and several magnetic tape units.

1959 A.D.

- The first Committee for the data system languages is formed, and **COBOL** is born (**C**ommon **B**usiness **O**riented **L**anguages). A new language is quickly developed at the **CODASYL** (**C**onference on **D**ata **S**ystem **L**anguages), so creating the first standard for business programs.



```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.  HELLOWORLD.
000300 DATE-WRITTEN.02/05/96   21:04.
000400*       AUTHOR  JOHN JONES
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
```

■ **Figure 2.119**
In the following 20 years, most programs are to be written in COBOL rather than in any other language.

1959 A.D.

- The **first Japanese commercial computer** working with **transistors**, the **NEAC 2201**, is introduced by the NEC in an exposition in Paris.

■ **Figure 2.120**
Picture of the first Japanese computer.

1959 A.D.

- After several years the **General Electric Corp.** sends 32 **ERMA** systems (**E**lectronic **R**ecording **M**achine **A**ccounting) to the Bank of America, in California. These had been required to solve the problem concerning the computation of cheques which have reached high amounts, as a consequence of a great growth in customers. ERMA uses the technology **MICR** (**M**agnetic **I**nk **C**haracter **R**ecognition), that to say it is able to read numbers written on checks.

■ **Figure 2.121**
A particular font is conceived for ERMA, to enable it to read more easily, due to its particular purpose.

1959 A.D.

- The IBM delivers to the United States Aviation the first four models of the first Computer entirely working with transistors, the **IBM 7090**. At the beginning of the 60's NASA's Saturn missile trajectories are to be computed an incredible number of times by the systems of the IBM 7090. This machine is able to execute 22,900 calculations/second. This parameter will create a standard unit of measurement in the industry: the number of floating points per second (**FLOPS**).

■ **Figure 2.122**
The IBM 7090 console and the complete machine.

1959 A.D.

- The **IBM 1620** will represent the first approach to computers for many students. The machine has a memory core oriented to character and with a capacity going from 20 to 40 Kb, where the limits of the words can be chosen by the programmer in order to obtain a limitless precision. It is provided with an arithmetic unit using a table of decimal research, instead of the binary adder.

■ **Figure 2.123**
The IBM 1620 with all its peripheral devices: the Disk drive, the tape unit, the memory unit, the card reader and the printing machine.

1959 A.D.

- The first mass-produced **Italian** computer, the **ELEA 9003**, begins. Adriano Olivetti takes part in ELEA 9003's mainframe project lead by Mario Tchou. Ettore Sottsass took care of its design and furnishings. He will receive the prize "Compasso d'Oro" for this reason.

■ **Figure 2.124**
ELEA 9003.

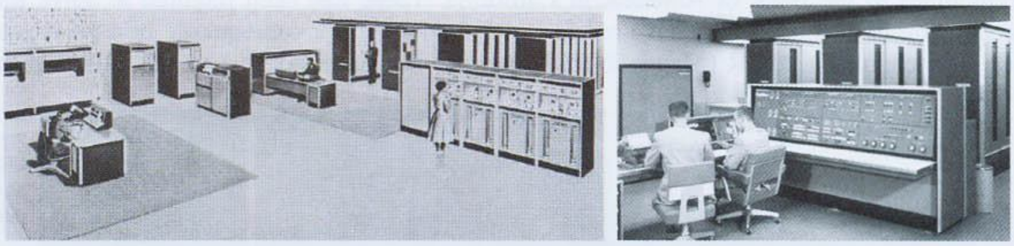
1960 A.D.

- The **Heatkit** puts its **EC1** on the market, for pedagogical purposes. It is the first analogue computer available for a wide range of public, due to its low cost, $400. The machine will become very popular among engineers and in schools.

■ **Figure 2.125**
EC1.

1960 A.D.

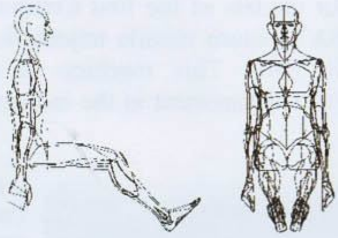
- **Remington Rand** projects the **LARC** (**L**ivermore **A**dvance **R**esearch **C**omputer), for scientific purposes.



■ **Figure 2.126**
The LARC will use 60,000 transistors.

1960 A.D.

- William Fetter, **Boeing**'s employee at that time, coins the word "computer graphics" for his **human factors cockpit drawings**.



■ **Figure 2.127**
The drawing is composed by seven different systems of joints, which allow one to animate it, change it, scale it, in accordance with different types of studies carried out for aircraft cockpits.

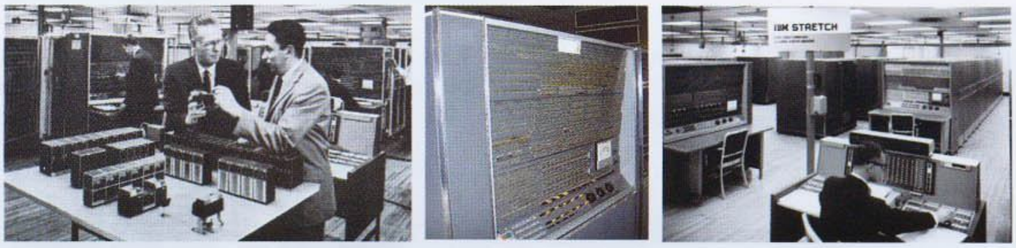
1960 A.D.

The DEC introduces the **PDP-1**, the first commercial computer equipped with a screen and a keyboard for inputs.

■ **Figure 2.128**
It costs between 125.000 and 250.000 dollars, according to its configuration.

1961 A.D.

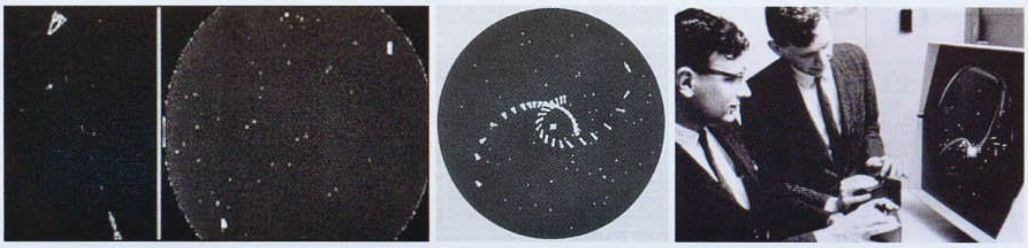
- The **IBM 7030**, called Stretch, is achieved and it works 30 times faster than the 704. It is the first supercomputer produced by IBM. The first 7030 is installed in Los Alamos in 1961. At the beginning it cost 13.5 million American dollars. The price will be lowered to 7.78 million dollars because of its efficiency, much lower than expected. For those customers who had already bought the machine, the price was lowered too.



■ **Figure 2.129**
Although it was much slower than expected, the supercomputer has been the fastest machine worldwide between 1961 and 1964.

1962 A.D.

- **Spacewar** is programmed by **Steve Russel** in 1962 using the central computer in Massachusetts Institute of Technology. The game simulates a sort of "space war", as the name shows, in a "one-on-one" duel. But the game will never be merchandised: in fact it only works on a university computer, the DEC-PDP-1, which costs 120,000 dollars. The game is very successful among those lucky enough to be able to purchase the PDP-1, one of these being a certain Nolan Bushnell, the future founder of Atari.



■ **Figure 2.130**
The graphics leave something to be desired, but nothing better had ever been seen.

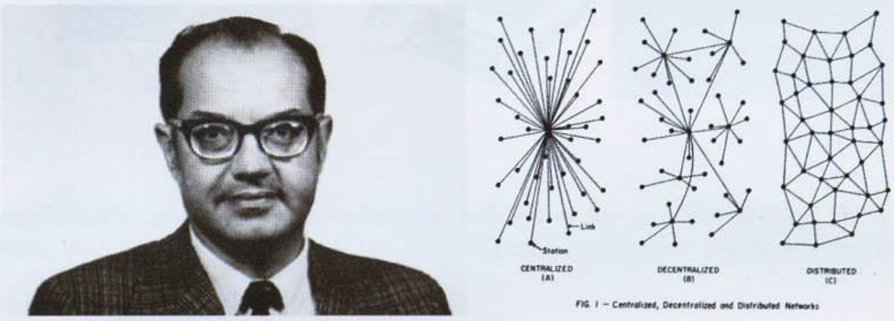
1962 A.D.

- The company Teletype issues **Teletype model 33**, composed by a keyboard and a tape unit. This kind of I/O will be used in microsystems until the beginning of the 70's and in mainframes until the 80's.

■ **Figure 2.131**
The terminal will represent the image of the computer in collective imagination for the following 15 years, also because it has been used in many science-fiction movies.

1962 A.D.

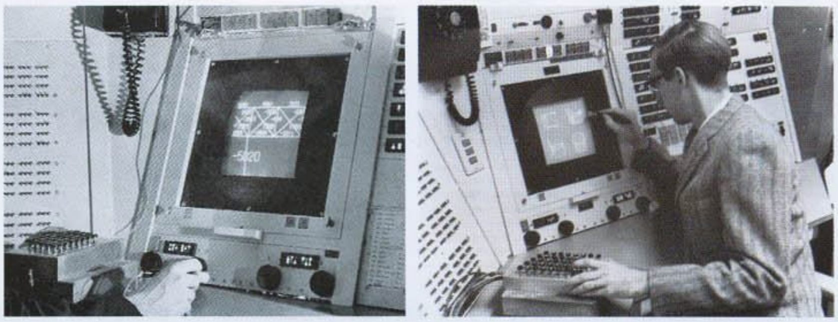
- The Rand Corporation issues **Paul Baran**'s report, entitled "On Distributed Communications Networks", which summarizes a research work, lead by Baran on request of the American Air force, and describes how American forces would manage to protect their communication systems in case of an attack. Baran outlines the "redundancy of connectivity" principle, and assesses the vulnerability of different models of telecommunication structures. The report advises a communication system without a real command centre. This would enable all the nodes of the net which have survived a possible attack, no matter where it happened, to re-establish the connection. In other words, any damage occurred in a part of the net would not affect the entire net, making it unusable, and the crash effect would be reduced to the minimum. Baran suggests the development of a national public infrastructure for data transferring, similar to the telephone one: **Internet**.



■ **Figure 2.132**
Paul Baran and his three kinds of Networks: centralized, decentralized, and distributed.

1963 A.D.

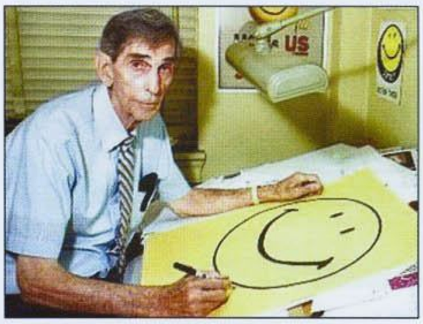
- **Ivan Sutherland**, at the MIT, achieves the first logic **graphic interface** using a light pen. In this way it is possible to write directly on the screen.



■ **Figure 2.133**
Graphic screen and light pen.

1963 A.D.

- **Harvey R. Ball**, American advertising agent, creates the **Smiley Face**, on the occasion of a campaign aimed at cheering up the employees of two insurance companies which had just merged.



■ **Figure 2.134**
His famous face composed by a yellow circle, two dots for the eyes and half circle for the smile is to become the basis for "cyberfaces" or "emoticons" later used in Internet.

1963 A.D.

- The **ANSI** (**A**merican **N**ational **S**tandards **I**nstitute) accepts the **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) with 7 bit code for information interchange. It is to become the standard in the whole world, still used today. Before that time each computer would use its own system of data representation. It was always necessary to use conversion tables to interchange information between different machines.

■ Figure 2.135
The IBM, instead, will use a different code in each following mainframe-system, called EBCDIC (Extended Binary Code Decimal Interchange Characters). IBM still uses this code, except for in personal computers.

Hex	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1-	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2-	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6-	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7-	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

8-	Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã	å	ç	é	è
9-	ê	ë	í	ì	î	ï	ñ	ó	ò	ô	ö	õ	ú	ù	û	ü
A-	†	°	¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø
B-	∞	±	≤	≥	¥	µ	∂	∑	∏	π	∫	ª	º	Ω	æ	ø
C-	¿	¡	¬	√	ƒ	≈	∆	«	»	…		À	Ã	Õ	Œ	œ
D-	–	—	“	”	‘	’	÷	◊	ÿ	Ÿ	⁄	¤	‹	›	ﬁ	ﬂ
E-	‡	·	‚	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ï	Ì	Ó	Ô
F-		Ò	Ú	Û	Ù	ı	ˆ	˜	¯	˘	˙	˚	¸	˝	˛	ˇ

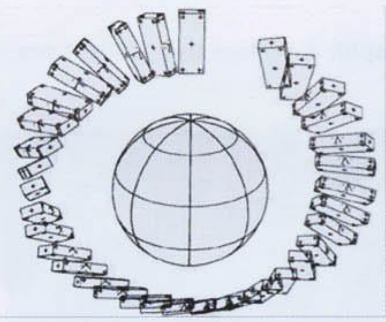
1963 A.D.

- **Philips** invents the **compact tape cassette**. This tape, apart from being useful in the consumer world, is to become a media used in many home-computers in the following 20 years.

■ Figure 2.136
It was originally composed by a certain quantity of BASF magnetic tape, held in a plastic protective shell. The tape had 4 tracks, allowing the recording of two stereo tracks.

1963 A.D.

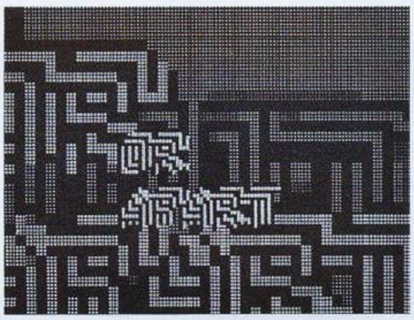
- **Edward Zajac** produces, at the Bell laboratories, one of the first **computer-produced** films, which demonstrates that a satellite can be steadily in orbit with any orientation. The film was entitled "A two gyro gravity gradient altitude control system".



■ Figure 2.137
Although he wasn't an animator or particularly creative, Zajac guesses the importance of his intuition. Thus, in the following year, he works to foster it among scientists and communication experts.

1963 A.D.

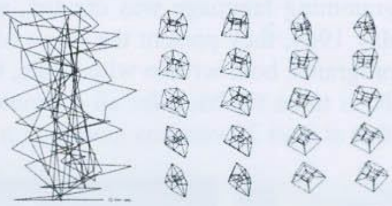
- **Ken Knowlton** develops a system called **BEFLIX**, a pad for the production of computer-animated images. It is performed with an IBM 7094 transistor-computer and a microfilm recorder Stromberg-Calson 4020. Its resolution is 252x184 pixels in eight grey colors. The output per minute cost is $500. It was possible to draw straight and curved lines, copy a screen area, fill an area, zoom and create fading effects.



■ Figure 2.138
Zajac had used FORTAN, a complicated language, for his aim. Knowlton instead develops an easier one, BEFLIX, forestalling the trend of approaching languages which make CG available for those who have not an engineering education.

1963 A.D.

- **Michael Noll** from Bell Labs begins the creation of his **Gaussian Quadratic works**.



■ **Figure 2.139**
Noll's researches on CG, like Csuri's ones, end up in the field of artistic experimentation, contributing to open up the Computer Art sector.

1964 A.D.

- IBM announces the **System/360**: it is the **THIRD COMPUTER GENERATION**. The 360 series is presented with programming languages such as Assembler, RPG (**R**eport **P**rogram **G**enerator) and COBOL. The operating system can be stored on tape (TOS) or on disk (DOS).

■ **Figure 2.140**
- An IBM System/360 workstation.
- Some IBM/360 integrated circuit cards. - Removable disk-pack replacement.
- IBM System/360's whole range, with models: 30, 40, 50, 65 and 75.
- IBM System/360's CPU.

1964 A.D.

- The smallest of the **IBM 360 Systems** is the **360/20**. It is equipped with a program storage capacity of 8 Kb and it has a microcode stored as firmware. As all the rest of 360 set, it works with removable disks (dispack) and magnetic tape systems.

■ **Figure 2.141**
The baby of the family: the 360/20 System.

1964 A.D.

- **Pier Giorgio Perotto,** Italian engineer, professor in the Politecnico in Turin, and leader of a team of engineers in Olivetti, achieves a machine called "**Programma 101**", forerunner of the PC. As large as a calculator, and used in Italian in the feminine, "la Programma 101", it will be later be named "la Perottina". It is marketed one year later. It is the first computing machine combining computer performances with the ergonomics of a calculating machine. Before its arrival, in fact, there were two possibilities: using very complicated and expensive computers, or settling for a desk mechanical calculating machine, able to execute the four standard operations. For data input/output, Perottina uses a magnetic card also usable as a permanent memory. It is provided with a streamlined machine language, constituted by 16 intuitive machine-code instructions: a sort of ante litteram Basic. Perottina is born as a pure research object, but unexpectedly becomes a success (40,000 specimens are to be produced). This convinces Olivetti to return to working on electronics and it is the beginning of a long period, where the Ivrea brand will hold a lead role in the international scene. The first **Personal Computer** was Italian!

■ **Figure 2.142**
Perottina's shape, often considered unsuitable by some managers in Olivetti, has been a masterpiece in ergonomics. As a consequence the company wins the Industrial Design Award.

1964 A.D.

- The language **BASIC** is born (**B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction Code). It is developed in Dartmouth by two professors, **John Kemeny** and **Thomas Kurtz**, together with the aid of many students. The first aim was purely educational, but a real programming language was created, adored by new generations because of its usability and simplicity. On the first of May 1964, they present the demo of their incredible new language, and of the time-sharing system, which enables two programs, both written with Basic, to be played at the same time on the mainframe of a GE. In the following years BASIC is to be the basis for all computer history, with all its versions available in every era. In 1970 it will be so advanced that at least 20 versions working on different mainframes will be present.

■ **Figure 2.143**
Besides being a really simple but high-level language, its advantage is having instructions fit both to be compiled in machine language and to be read directly by the source code.

1964 A.D.

- Because of the speed of 9 megaflops, and 9 million operations in floating-point per second, **Control Data Corp**. claims that its **CDC 6600**, projected by **Seymour Cray**, deserves to be considered as "the first successfully trading supercomputer". This computer is provided by 10 peripheral processors, known as pre-processing units, sending data to the central processor, that is the CPU. The speed of the processor is extraordinary: 3,000,000 instructions per second (3 MIPS).

■ **Figure 2.144**
The CPU of the CDC 6600 and some workstations.

1964 A.D.

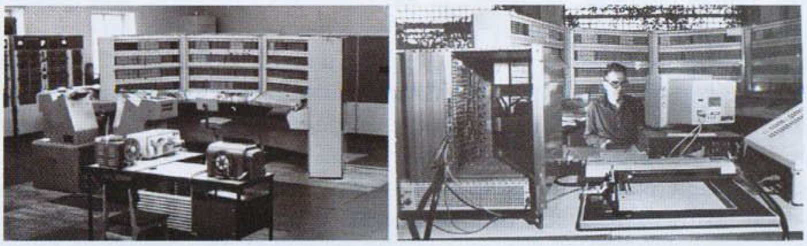
- Thanks to a project carried out by both **IBM** and **General Motors**, the **CAD** (**C**omputer **A**ided **D**esign) is developed, paving the way for technical and design planning via computer.

1964 A.D.

- The first **LAN** (**L**ocal **A**rea **N**etwork), local network used to connect several computers without passing through telephone lines, is developed in the **Rank Zerox Palo Alto Research Center**.

1965 A.D.

- **Sergei Alexeevich Lebedev** develops the **BESM-6**, the last supercomputer in its generation by the Russian company ITMiVT, starting with the BESM-1 in 1953.



■ **Figure 2.145**
This machine is provided with a memory processor of 48 bits, 9 MHz and 192 Kb. 350 specimens will be produced. Its production will stop in 1987, and its successor will be ELBRUS.

1965 A.D.

- The first computer with vector architecture is produced: the **ILLIAC IV**. The **ILLIAC IV** is one of the most famous supercomputers, but of ill repute too. It has been the last of a succession of computers used for Illinois University research. The ILLIAC IV project is based on a massive parallelism with 256 processors, developed to manage large amounts of data: the same idea vector processors are based on. The machine has been presented in 1976, after ten years of work, when it was already overdue, very expensive and its performance was lower than machines such as Cray-1. Although ILLIAC IV did not succeed as desired, the study of its faults and architectural failures brought to the development of parallel architecture types. This prompted the creation of massively parallel machines, like CM-1 and CM-2 Thinking Machines.

The employment of processors arranged in matrix configuration is one of the key ideas for parallel machines and represents a main step in the scientific world.

■ **Figure 2.146**
The CDC STAR-100, the TI ASC, and the famous Cray-1 are high-performance machines, based on ILLIAC's idea of single vector processors.

1965 A.D.

- Intel's chairman, **Gordon Moore**, and his famous law. In 1964 the chips produced and sold were still few. Despite this and the sparse information and data available to him, the engineer guessed the power of microprocessors would have grown very quickly. This idea has been clarified and changed in a law: the number of transistors used in a chip doubles every 18 months, and, with downsizing, the reduction in the size of microprocessor elements and the costs to produce and merchandise chips, lower proportionally. Just to have an idea, the most complex and evolved chip in 1965 was formed by 64 transistors, while 42 millions are present in a Pentium 4. The effects of Moore's law are quite evident on the market.

■ **Figure 2.147**
A computer costing 3,000 euros today will cost half the amount next year, and will be outdated the year after.

1965 A.D.

- The first **optical fiber** is used in an **IBM** card reader.

1965 A.D.

- The **DEC** introduces the **PDP-8**. It uses transistor-circuit modules. While many companies develop machines which are increasingly large and fast, Digital Equipment Corp. presents its first minicomputer. The PDP-8 has a set of instructions, a rough micro-language (operating system) and excellent interface faculties. For this reason PDP-8 is to be massively used as a checking system for processes, included the interfacing to telephonic lines for time sharing systems.

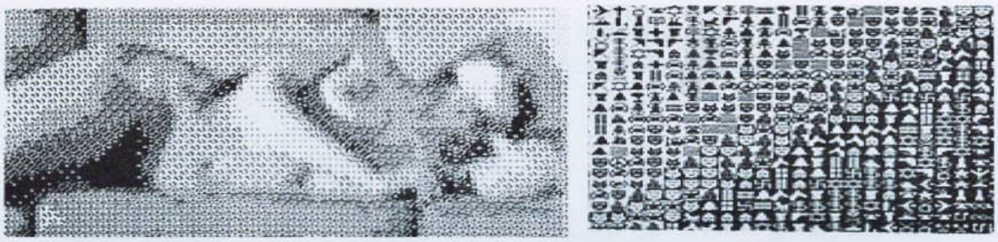
■ **Figure 2.148**
The PDP-8.

1965 A.D.

- At the **Utah University**, the **science department** is founded. From now on it is to be one of the most profitable places for CG development. Many of the most important algorithms and theoretical layouts are to be developed, such as Hidden surface algorithms (Romney, Warnock Watkins), Z-buffer (Catmull), Texture mapping (Catmull, Blinn), Shading (Phong, Gouraud), etc...

1966 A.D.

- **Ken Knowlton**, researcher in Bell Labs, focuses his attention on using computers to create images. At that time it was not possible to change pixel intensity on a screen. These were just turned on or off. Knowlton's aim was managing to display real images via computer, not only graphic drawings performed with cages of lines, typically CAD images. For this reason, his purpose was finding a way to perform light and shadow effects, a basic technique for any image. He guessed two important facts. The first one was that each image can be divided into smaller elements, so that each one can be treated as tiles in a mosaic. The second idea, more acute, was that alphabetic characters have different luminous properties according to their density. For example a sequence of 8 is going to be darker than a series of commas. Thus, from now on, the final product is that an image can be reproduced as a mosaic composed by letters, both by a printing machine and any cathode screen. **Graphic raster** is born this way. Knowlton is definitely the inventor of pixels!



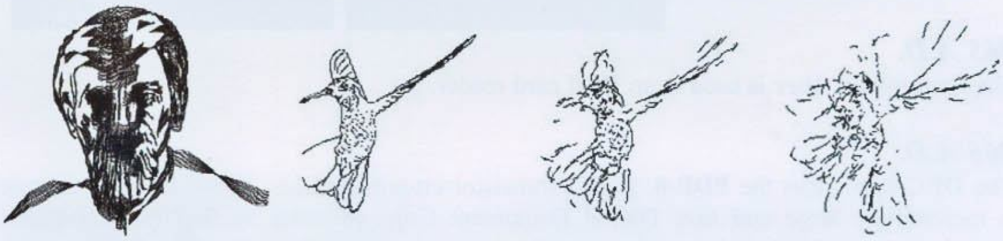
■ Figure 2.149
Thanks to this "study on perception", carried out in Bell Labs, Ken achieves one of the most famous images in the CG pioneering stage. It is to be used several times as a benchmark.

1966 A.D.

- **M.A.G.I.** (**M**athematica **A**pplications **G**roup **I**nc.) was founded for military purposes with the aim of setting and evaluating the danger of nuclear radiations. Software based on the **ray-casting** idea was created, able to draw radiation paths from the source to an external environment. Afterwards, this software, named **SynthaVision**, had been adapted for CGI, where light beams were used instead of radiations. It became one of the first programs using **ray-tracing** for many generations of virtual images. Thanks to this software, shaping simple geometries and combining them with common operations was possible. This shaper is combined with one of the first rendering engines able to create high quality images.
Basically, the idea behind ray-casting is using rays, one for each pixel, which having left the camera, proceed until they are blocked by something. Using the properties of materials, and considering the illumination of the scene, this algorithm is able to compute the shader of the object encountered along its path. Thus it is clear that only direct illumination can be simulated with this system, because, after the first collision, the ray ends its life.

1967 A.D.

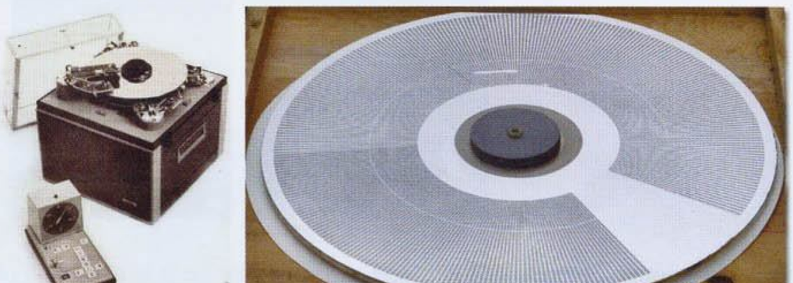
- **Charles Csuri**, arts school professor in Ohio's University, becomes known thanks to animations fulfilled manually, then digitized and finally manipulated in time thanks to an IBM 360. Although Csuri's animations weren't real animations, but rather transformations of graphic elements in time, they highlight the potential of computer graphics.



■ Figure 2.150
Photograms of two important animations by Csuri, Sine Curve Man and Hummingbird.

1967 A.D.

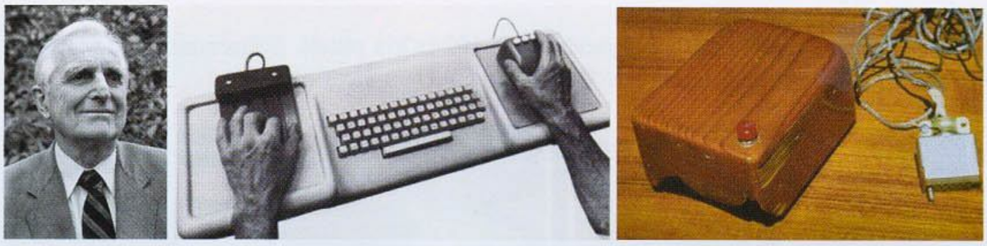
- The **Ampex HS-100** disc recorder is produced. It is the first machine able to reproduce images in **slow-motion** and **replay**. Due to memory demand, it is used only for short recordings aimed at showing its characteristics. It is a rigid disk used for analogue video recording rather than digital data.



■ Figure 2.151
It spun at 1,800 rpm, and it could record for a maximum for 30 seconds.

1967 A.D.

- **Douglas Engelbart**, from Stanford Research Institute, receives the patent for his "X-Y Position Indicator" for display, today known as the **mouse**. Engelbart had thought about this device for ten years. He will publicly show the mouse one year after on Online System, showing videoconferences and hypermedia too. Those inventions hadn't been projected to earn money or create a product, but rather to help people work together more easily and make the world better. But his idea of managing hypertexts, word processors, keyboards, mouse and windows was absolutely untimely and too expensive to fulfil, thus it had not been seriously considered. But Steve Jobs, from Apple Inc., later concentrated on this point, on Rank Xerox's request, and went ahead developing Lisa, which was still expensive, and Macintosh, expensive but affordable, based on the ideas he had seen in the Palo Alto Research Center (PARC).



■ **Figure 2.152**
Engelbart invents the mouse, a pointing device, in the Stanford Research Institute. He was working on hypertextual systems: clicking with the mouse on certain points in the screen, another text would appear, with more information.

1967 A.D.

- **IBM** achieves the first 8-inch **Floppy disk**. David Noble begins to develop the first memory system on flexible disk, a floppy, in order to record the initial computer check program on it. The ICPL (**I**nitial **C**ontrol **P**rogram **L**oad) is used to start the computer, and two years after this "strange object" will be used on the IBM System/370.

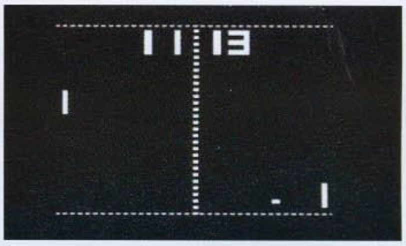
■ **Figure 2.153**
The term "IPL" will be used on all the following IBM processors. It is referred to as the starting or restarting procedure, in opposition to the term "Boot" used for other computers.

1967 A.D.

- The **Fairchild Semiconductor Inc.** Company develops the first 256 bit (yes, really, bits!) chip of **RAM** memory (**R**andom **A**ccess **M**emory). The chip contains more than one thousand transistors.

1967 A.D.

- The first functioning version of the **electronic ping-pong**, carried out by **Ralph Baer**, Bill Rush and Bill Harrison, is presented.



■ **Figure 2.154**
Ralph Baer is the game's true father: in fact Atari will rename Pong, copying his idea. Shortly after he will create the Magnavox Odyssey game console.

1967 A.D.

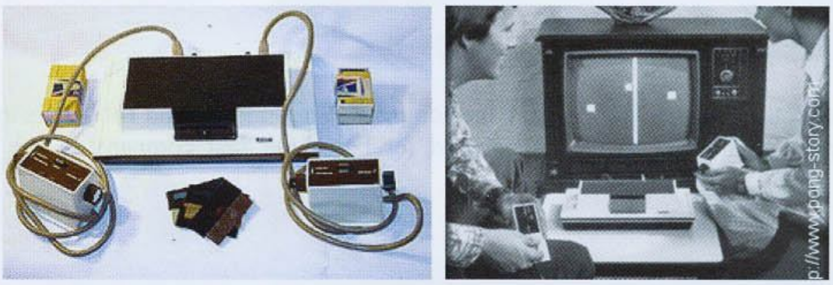
- The Utah University computer department, lead by the professors **David C. Evans** and **Ivan Sutherland**, is specialized in the production of 3-dimension computer images. They are to found **Evans & Sutherland** in 1967.

1968 A.D.

- A ruling of the Federal Information Processing Standard promotes the employment of **six digit dates** (yymmdd), for information exchange...sowing the seed of the Millennium Bug!

1968 A.D.

- **Magnavox Odyssey** is produced. It is the first marketed videogame console, carried out three years before Atari's PONG. Odyssey had been projected by **Ralph Baer** which had worked on the prototype and achieved it in 1968. By its fans the prototype is known as the Brown Box. Contrarily to many following consoles, Odyssey was an analogue, and not digital, machine. The fact that it was equipped with basic sprites surprised many. The machine was provided with rough graphics as well as a controller for interacting with games. Unlike typical systems this console was powered by batteries. It was the **first videogame console** in the world.



■ **Figure 2.155**
The Odyssey and its variations (except for Odyssey 2) weren't provided with sound systems. It was not unusual at that time. PONG too had no sound system.

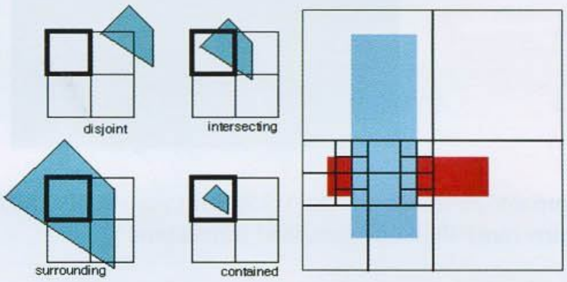
1968 A.D.

- The **Tektronix 4010** is produced. Its resolution in graphic modality is 1024x768 pixels. Subsequently the larger-sized 4014 model is produced. Because of this characteristic, it is to be used in CAD applications.

■ **Figure 2.156**
The first two images are referred to Tektronix 4010. The other two, to its older brother, the 4014.

1968 A.D.

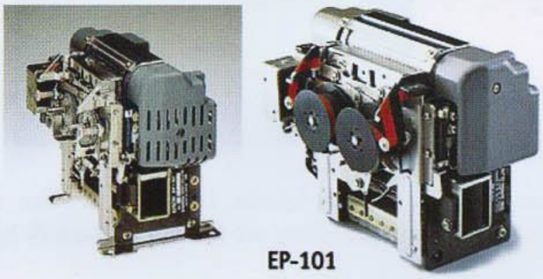
- **Warnock's algorithm** is introduced. Thanks to him, it will be easier to handle the viewing of hidden areas in 3D applications.



■ **Figure 2.157**
Algorithm scheme.

1968 A.D.

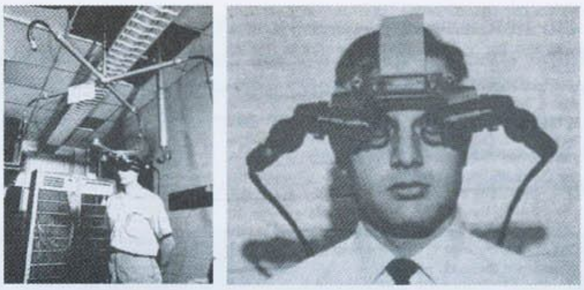
- **Epson** invents the **EP-101**, the first dot matrix printer. The new machine is developed on request of the Seiko head office, which requires small devices for use in the Olympic Games of Tokyo, for time recording.



■ **Figure 2.158**
Another 4 years will be necessary for these printing machines to become mass-produced.

1968 A.D.

- **Sutherland** develops its first **VR** (**V**irtual **R**eality) system. Because of hardware which is still not powerful enough, the project remains unfinished, but not abandoned. The project will be resumed later also thanks to Charles Seitz, an old Harvard student.



■ **Figure 2.159**
Prototype of the first "3D" glasses.

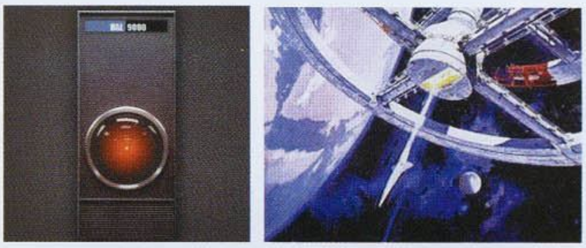
1968 A.D.

- **Robert Noyce**, **Andy Grove** and **Gordon Moore** leave the company Fairchild to found **Intel**. It is to become a leader in its market range, and most of all it is to be remembered for being the first company in the world producing microprocessors.

1968 A.D.

- **Arthur C. Clark** presents HAL: the computer used in the future, for the film **"2001: A Space Odyssey".** It is based on the idea of artificial intelligence suggested by I.J. Good and Marvin Minskey. It seems the name HAL comes from the three letters preceding IBM's acronym.

Stanley Kubrik, the film director, contacted Sir Clark because he needed a good science-fiction story to create a similar film on. So the novel and the movie started and grew together, making for an absolutely unique and ingenious collaboration among different media, at least for the period it took place in.



■ **Figure 2.160**
Clark himself had already published his ideas about using orbit satellites for communications.

1969 A.D.

- The **PASCAL** language is born thanks to **Niklaus Wirth**, Swiss scientist.

1969 A.D.

- Datapoint was a Japanese company with its headquarters in the USA as well. Its developers project a simple combination of a processor and a CPU. Both Texas Instruments and Intel are invited to build this device using a single integrated circuit. Intel succeeds but it is immediately evident that the processor is 10 times slower than Datapoint had expected. Afterwards the project was blocked, due to Datapoint's failure. In spite of this, Intel makes Marcian Ted Hoff leader of a team of technicians to continue working on the project. In 1971 they invent **Intel 4004**, a 4-bit CPU. A whole computer contained in a chip. It is publicly issued on Electronic News on the 15th of November 1971. This will start an electronic revolution. Intel had something truly revolutionary in its hands, able to offer, in a few centimetres, all ENIAC's computing power. But it is said they did not understand this right away, thinking it would have just been a way of selling more RAMs, their special product at that time. The initial price of the 4004 was 200 dollars, and its possible applications were many. It will be Wayne D. Pickette to prompt the original idea of placing a whole computer on a single chip only. The same year Intel announces a new RAM chip with 1 Kb capacity. It was partially successful, but later on, Intel 8008 and 8080 got more attention, using 8 bit-long words, finally one byte.

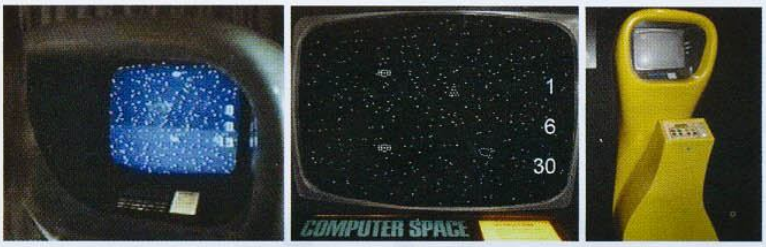
■ **Figure 2.161**
The Intel 4004 is one of the most sought after chips among collectors. The most precious ones are the golden and white-color 4004 models, because the so-called "grey traces" are visible on the white area (the original package). In 2004 these chips have been sold on e-Bay for € 400.

1969 A.D.

- Gary Starkweather, from Webster's Xerox research labs (NY), shows it is possible to use a **laser ray** for **printing**. This will put Xerox on the right path. But Xerox fails to market this invention, thus being obliged to sell the idea to HP, which is to become the biggest supplier of printing machines in the following years.

1969 A.D.

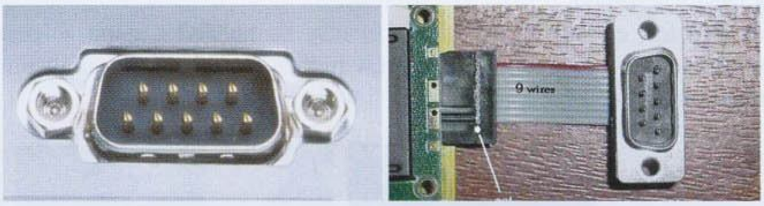
- **Nolan Bushnell**, later the founder of Atari, creates **Computer Space**, an arcade videogame using a raster type TV screen. It does not succeed on the market, but it is enough for us to consider Bushnell as the father of arcade videogames.



■ **Figure 2.162**
Arcade version of Computer Space.

1969 A.D.

- The serial interface **RS-232-C** has by this time become a worldwide standard. It allows communication between any kind of computer and a peripheral. Information is still sent one bit at a time. For this reason it is called "serial port".



■ **Figure 2.163**
The opposite of "serial" is "parallel": in this case data is sent at the same time on several threads, for example 8, 16, 32. This is the reason it will earn its name.

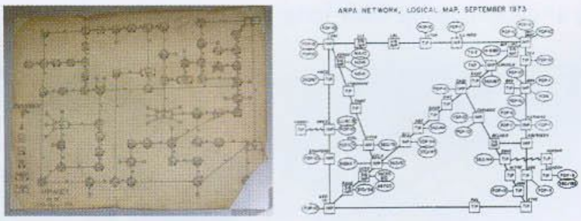
1969 A.D.

- **Ritchie** and **Thomson**, from Bell Telephone, start working at their new operating system. The idea is that of an operating system aimed no longer at multiple users, but at one user only. For this reason they call it **UNIX**. The first version will be run on the PDP-7, produced by Digital Equipment Co. and will be achieved the same year. The program is written in assembler PDP-7. UNIX is to become powerful among operating systems.

■ **Figure 2.164**
UNIX' fathers.

1969 A.D.

- The US Defence Department orders **ARPAnet**. As Bolt, Beranek and Newman builds IMPs, which are Honeywell DDP-516 minicomputers with 12 K of memory used as a gateway, several computers of different models are linked. These are the nodes of the rising ARPAnet. On the 2nd of September the first node is linked (UCLA); the second (Stanford Research Institute, or SRI) is linked on the 1st of October. On the 1st of November also the IBM 360/75 at UCSB (University of California Santa Barbara); in December the fourth node is linked, the one at Utah University, a DEC PDP-10.



■ **Figure 2.165**
ARPAnet schemes. It continues evolving in Government and University environments. From 1974, when the TCP/IP transmission standard starts to be used, the project will be renamed Internet.

1969 A.D.

- The **first remote login** between two computers in the USA is carried out. It can be considered the **first e-mail**, meaning a textual message exchanged between two computers. Professor **Leonard Kleinrock**, considered by many the father of Internet, manages to make his computer in California University in Los Angeles communicate with another one, located elsewhere, in Stanford's research centre, near San Francisco. But a few seconds after the two-letter-long message had been sent, the computer crashes.

■ **Figure 2.166**
Professor Leonard Kleinrock.

1969 A.D.

- **Linus Torvalds**, future founder of the free operating system **Linux**, is born in Helsinki.

■ **Figure 2.167**
Torvalds had to study the new Intel 80386 processor. On the machine he used the software running was OS Minix, a small commercial UNIX offered with special terms for education purposes. Linus wanted to improve Minix. Not managing to do it, he developed his own kernel.

1969 A.D.

- On the 1st of May 1969 **Jerry Sanders** and seven of his friends found the **AMD** (**A**dvanced **M**icro **D**evices). It is to become a multinational company producing semiconductors (microprocessors, memory devices, support circuits for telecommunications and computer applications) in Sunnyvale, California.

■ **Figure 2.168**
The Company logo.

1970 A.D.

- **IBM** announces a new series of mainframes, the **IBM System/370**, an evolution of the 360 systems. The development of mainframes, vast IBM systems, is continuously growing, supplying machines for data computing to companies worldwide. The 370 series differs in the management of the virtual memory, a new event indeed.

■ **Figure 2.169**
IBM System/370 (CPU, disk packs and console).

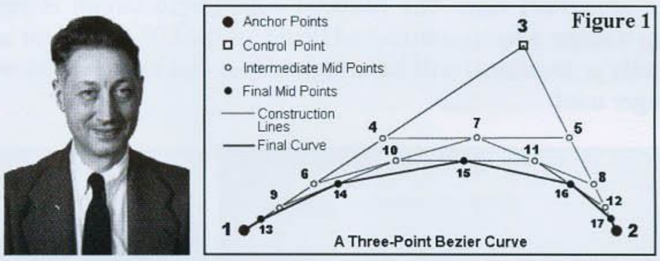
1970 A.D.

- For the first time ever a rendered image carried out via computer, is used in a movie: **"The Andromeda Strain"** (in the lab's rotating structure).

■ **Figure 2.170**
Movie's poster and a film image.

1970 A.D.

- **Pierre Bezier** develops his famous **Bezier freeform curve** during his job in Renault. At the beginning the NURBS are used in CAD proprietary software. It is necessary to wait until 1989 to use the NURBS real-time, thanks to the power supplied by Silicon Graphics workstations. In 1993 the first commercial software patterned on NURBS, NòRBS, produced by CAS in Berlin, is created.



■ **Figure 2.171**
Pierre Bezier and a chart describing his theories.

1970 A.D.

- **Ken Thompson** and **Dennis Ritchie**, having just undergone the interruption of the **Multics** project, after having developed the operating system UNIX, write another programming language, called "B". It is said this name was chosen because another language named "A" already existed. In 1972 Dennis Ritchie will rewrite "B", renaming it "**C**".

■ **Figure 2.172**
Ken Thompson and Dennis Ritchie.

1970 A.D.

- The first **IBM** 8" **Floppy Disks** with 130 Kb capacity make their debut.

■ **Figure 2.173**
IBM floppy disk.

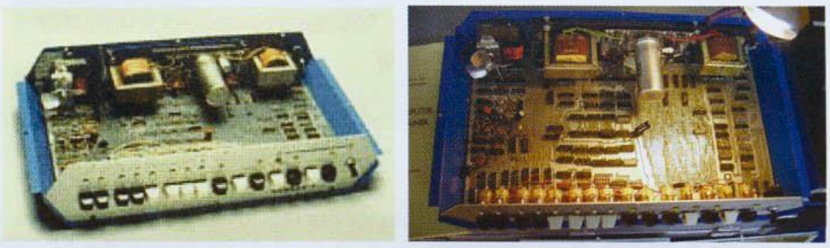
1971 A.D.

- **Henri Gouraud** invents **Gouraud shading**. It represents the shading model by which each polygon is shown with several colors. The result of environment illumination overlying the object's surface is computed on each vertex of each polygon. The color of the internal surfaces is set by an interpolation of values obtained on the vertex. This technique produces realistic objects with very smooth surfaces and requires a lot of calculating power, at least it was so at the time. Nowadays everything is performed in real time.

■ **Figure 2.174**
Gouraud shading is the model present nearly in all 3D accelerators containing rendering hardware functions, as well as in 3D software and their viewports.

1971 A.D.

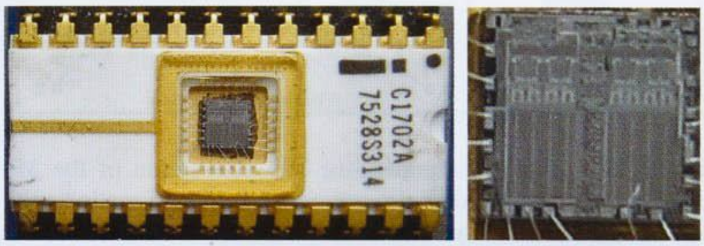
- One of the first personal computers is produced, the **KENBAK-1**.
It uses 130 IC with 256 bytes for memory and 8-bit words. It computes 1,000 instructions per second. Cost: $750.



■ **Figure 2.175**
Kenbak-1.

1971 A.D.

- **Intel** creates its first **EPROM**, acronym for **E**rasable **P**rogrammable **R**ead **O**nly **M**emory. It is a random access memory, erasable thanks to ultraviolet rays. The EPROM's integrated circuit is equipped on the upper side with a quartz glass window. Putting it under a lamp emitting a UV ray of the UV-C type for some minutes, all memory content is erased, and all the Bits (cells or locations) will have high values and the EPROM will be reprogrammable. EPROMs are now outdated and no longer used.



■ **Figure 2.176**
EPROMs have been replaced by EEPROMs (Electrical Erasable Programmable Read Only Memory), which can be erased electrically without using UV rays.

1971 A.D.

- In this year the **first microprocessor in the world** is obtained, thanks to the request of the Japanese Company Busicom who demanded the development of the electronic part of a desk-calculating machine.
Ted Hoff redesigned the whole circuit using 1 chip instead of 12. The chip contained the entire central computing unit (CPU), in addition to the RAM and ROM memories. The first microprocessor basic project was adjusted by **Ted Hoff** and **Stan Mazer**, while the task of transforming this idea into a working machine was committed to **Federico Faggin**. The electronic accomplishment of the scheme carried out by **Faggin** brought to the completion of the first microprocessor: the **Intel 4004**. Faggin, Hoff and Mazer will have a place of honour in the National Inventor's Hall of fame in USA for their invention.

■ **Figure 2.177**
- Federico Faggin.
- The first microprocessor: Intel 4004.
- The internal circuit of the Intel 4004.

1972 A.D.

- **Nolan Bushnell** founds **Atari**, while **Al Alcorn** enters Atari as first engineer and develops Pong. Magnavox accuses Atari because of the resemblance with its own game, Odyssey. The court favours Magnavox and Atari will have to pay a user license for its Pong.

ATARI®

■ **Figure 2.178**
The first Atari logo and its founder, Nolan Bushnell.

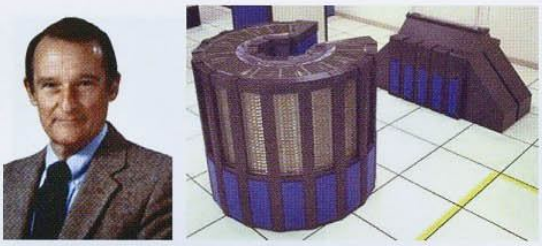
1972 A.D.

- The commercial "**at**" or "**@**" is born. The first experiments of sending messages via Internet are carried out. E-mail had not been thought of as the primary purpose of the Net, and no one among the founders had thought about it, because the Net was born to connect computers, not people. E-mail was introduced almost casually, in spare time. Thus it is necessary to choose a symbol separating, in a user address, the user name from the computer name on which the writer is working. **Ray Tomlinson**, working for Bolt, Beranek and Newman, sitting in front of his telex, notices the "at" (which was located on top of the "2" on that model, Model 33). Spontaneously he thinks it would be nice to write the addresses as "user AT computer", and this is the reason why e-mail addresses use this symbol.

■ **Figure 2.179**
The first message he sent, and the first mail sent, was "QWERTYUIOP".

1972 A.D.

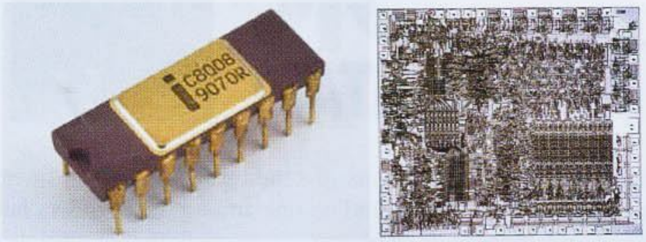
- **Seymour Cray** founds **Cray Research, Inc.** In the 80s it is to become the biggest company producing the most powerful supercomputers. Seymour Cray was an American electronic engineer and businessman, as well as an important character in the computer world. He started of in the field of computers with ERA (Engineering Research Associates); he worked at ERA 1100 series, which later became UNIVAC. After commercial ups and downs, Cray left ERA for the CDC (Control Data Corporation), where he started to develop particularly fast computers, pointed towards scientific purposes. His first creation was the CDC 1604, followed by the CDC 6600 and the CDC 7600. In 1972 he decided to go solo, founding Cray Research Inc., which gave way to the Cray series. The first one, Cray-1, with an 83 MHz processor had been installed in Los Alamos National Laboratory in 1976 for 8 million dollars, and became known as one of the best supercomputers in history. Afterwards Cray X-MP was created.
The following one, Cray-2, in 1985, renamed Bubbles, contained up to 8 244 MHz processors, 12 times faster than the best Intel microprocessor of that period, but produced enough heat to provoke third-degree burns: it has been the most heated processor in history so far. To cool it a Supercooling Unit had been used, using liquid freon at -271° and fluorinert, very expensive, derived from freon and a non-conductor. In 1988 Cray Y-MP was produced, a supercomputer which had a fair amount of success. In the 90s, the arrival of computers and companies developing new technologies put a strain on Seymour Cray's supermachines. The Cray-3 was a big flop. Technical problems and delays excluded this product from the market. In 1996 Cray Research was purchased by Silicon Graphics for 764 million dollars, and sold in turn to Tera Computer in 2000. After this transaction it is to be renamed Cray Inc. **Seymour Cray** dies on the 5th of October 1996 in a car crash in Colorado.



■ **Figure 2.180**
Seymour Cray and his second supercomputer, Cray-2.

1972 A.D.

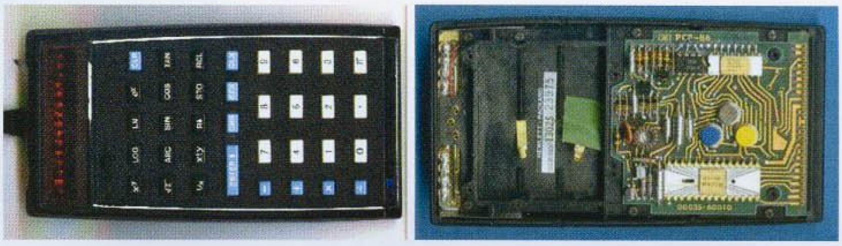
- **INTEL** markets the microprocessor Intel **8008**, with a speed of 200 KHz. It is to be noted that 8008 is exactly twice the Intel 4004, produced previously. It is, in fact, the 8 bit version of 4004. It contains 3,500 transistors based on 10 micron technology. It has been the first processor able to distinguish all the alphabetic characters (letters and numbers). The speed is 300,000 instructions per second and it is able to address 16 Kb of memory. The processor is to be immediately acknowledged by all the producers of small computers. Clock speed: 500 KHz and 800 KHz. Addressing: 16 Kb of effective memory.



■ **Figure 2.181**
- Intel 8008 microprocessor.
- Intel 8008 processor interior.

1972 A.D.

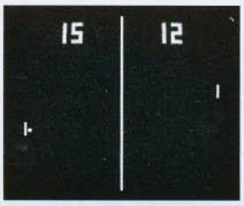
- **Hewlett Packard** issues the first pocket scientific calculator: the **HP 35**. This machine will brush away all the rulers used until that moment for calculating. It included already many features of the following one, the HP 9100A, and managed to change the calculation methods of mathematicians and engineers. It contained 8 ICs and it was sold for $395, a large amount for that time.



■ **Figure 2.182**
HP 35.

1972 A.D.

- **Nolan Bushnell**, **ATARI** founder, and **Al Alcorn**, his first engineer, demonstrated a strange TV in a tavern in Sunnyvale, California. They switched on the device and a rough playground simulating ping-pong appeared on the screen. Thanks to two handles they started what can be considered as the first videogame match in a bar. The program is called Pong and it was initially composed by two bars simulating paddles, a circle for the ball, a line for the net, and two counters for the score. This machine is already equipped with a device for coin insertion. Ten years later those games in pubs and bars will produce 5 billion dollars per year.



■ **Figure 2.183**
A display of the famous Pong, invented in 1968 by Magnavox Odyssey.

1972 A.D.

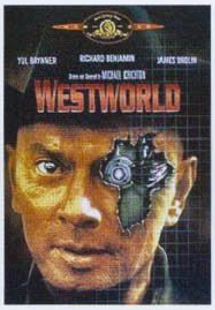
- **Dennis Ritchie** develops the "**C**" language in Bell Labs. It has been called so because it is an evolution of the language Ken Thompson had used for the first UNIX operating systems.

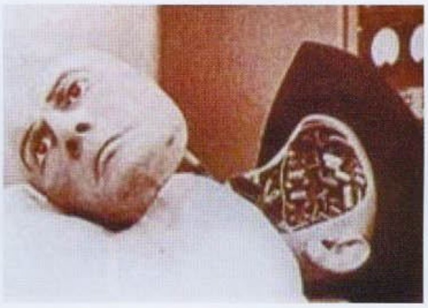
■ **Figure 2.184**
Original handbook cover.

1973 A.D.

- Several **digitalized 2D images** are used for the first time in the movie "**Westworld**" (the robotic stare, and the android's infrared gaze).

"Delos offers whoever wants to spend an unusual holiday, three computerized parks, where anybody can amuse themselves acting the part of an imperial senator, or a gladiator in Romamundia, becoming king or knight in Medioevonia, getting dressed as a pioneer or a gunman in Westernlandia. Two friends choose this last option to spend some days in. They are especially interested in challenging the Black Kinght. The walk-on actors of the theme parks are, in truth, very sophisticated robots, almost indistinguishable from human beings, and the guns are set to distinguish humans from robots, so as to avoid shooting at live targets. One day though, the robot control system seizes up and robots start to spread death among tourists. A human is killed by the Black Knight, who sets off in pursuit of the other one in a sequence which surely has inspired James Cameron in his creation Terminator."



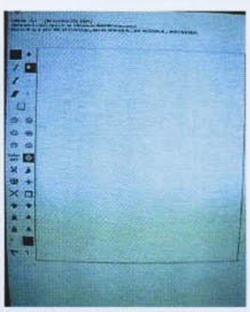


■ **Figure 2.185**
- Film poster.
- Westworld film sequences. The first one shows the robot's infrared vision, and the second one the robot's zooming vision.


1973 A.D.

- Some researchers in **Xerox PARC** decide to develop a computer fit to be used for research, and project an **experimental PC** named **Alto**, which uses a **mouse**, the **Ethernet**, and a **graphic user interface** (GUI). Not many specimens were sold, as it was expensive, but it represented the stepping stone for the development of for personal computers with a graphic interface. Alto had been the result of the combined efforts of Ed McCreight, Chuck Thacker, Butler Lampson, Bob Sproull and Dave Boggs. They were trying to create a device small enough to comfortably get into an office and powerful enough to back high quality operating systems and a graphic display. Their aim was to supply a personal computing device able to satisfy individual needs and enable them to share on the network with other similar users. In 1978 Xerox gives away 50 Alto computers to Stanford universities, to Carnegie-Mellon and to MIT. These machines will immediately be integrated in research communities soon becoming a model to be compared to for other computers. Alto PC is composed of four parts: the graphic display, the keyboard, the graphic mouse and the unit containing the processor and disk memory. It is perfectly achieved and it is sold for $32,000 (1979's value). Certainly not the price for a personal computer. It must be noted that we are talking about 1973. In an instant, starting from the first rough microprocessor, we catch a glimpse of what might be the future of individual computing. Ideas are accomplished with extreme simplicity. For the time being, the efficiency-to-price ratio is not feasible, but we do have to give credit to Xerox designers for having been able to achieve an exceptional machine for those times, and for having had a fine nose in understanding the services to be included in a personal computer. In 1973 this is still science-fiction, although perfectly achieved. For several years after this endeavour no graphic interfaces will come up. Xerox computers will not be able to spread to the market as they are very expensive and thus not available to a wide range of users.



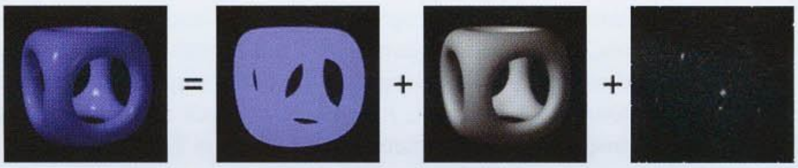


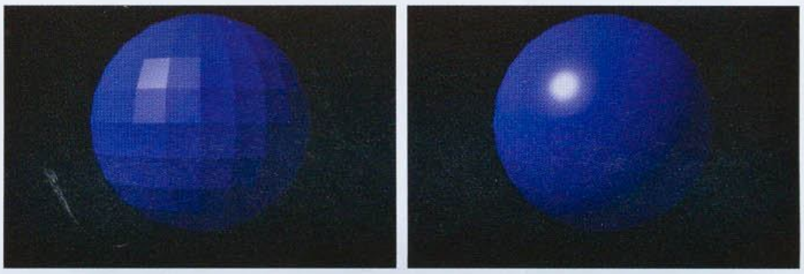
■ **Figure 2.186**
- Xerox ALTO Personal Computer.
- The revolutionary vertical screen and Xerox ALTO's graphic interface.
- The Xerox ALTO drawing program.

1973 A.D.

- **Bui-Toung Phong** develops the **Phong** shading method at the Utah University. It is a complicated technique, which computes light effects no longer on polygons, but on every single pixel of the object to be rendered. Results obtained with Phong Shading are higher when compared to Gouraud Shading, instead the number of calculations needed grows excessively. It combines the three principal elements: diffused color, specularity and environment color.

■ **Figure 2.187**
A graphic representation of Phong's equation.
- Comparison between the first graphic method and the Phong shader.





1973 A.D.

- The Japanese company **SHARP** develops **LCD** technology (**L**iquid **C**rystal **D**isplay). It remains the leader in this field for many years. This technology will then lead to flat screens for laptops. It is grounded on the principle by which crystals can be oriented if influenced by an electric current.

■ **Figure 2.188**
A modern cheap calculating machine display.

1973 A.D.

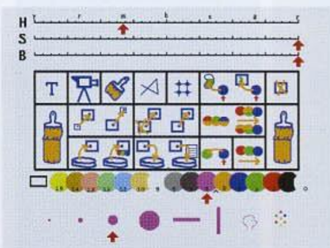
- The **BAR code** is born. It is composed by two black lines of different width which can be read by an appropriate scanner. Nowadays almost all products worldwide have this code on their label. Thanks to this system, automatically reading, recognizing, codifying products and finding the related price has been made easier. Bar codes are a group of graphical elements with high contrast, arranged in such a way that they can be easily read by a sensor and deciphered by an integrated circuit. On the 26th June 1974 in a supermarket in Troy, Ohio, the first product, a chewing-gum packet, was sold by using a bar code reader. The chewing-gum packet is now kept in the Smithsonian Museum of American History.

■ **Figure 2.189**
Some modern bar codes.

1974 A.D.

- **Dick Shoup** develops the first painting software called **Superpaint**, running on a Xerox PARC.

■ **Figure 2.190**
Software toolbar.



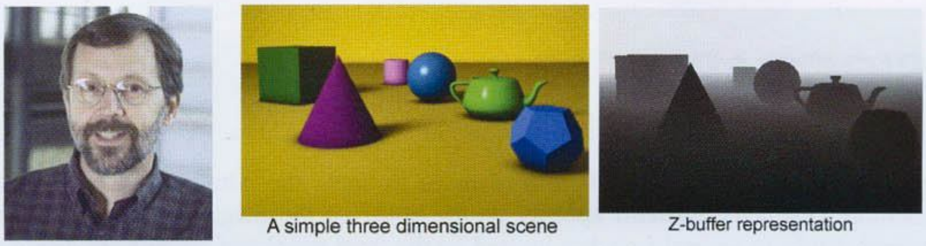
1974 A.D.

- **Peter Foldes**, from the National Research Council, Canada, produces the short film **Hunger**. He wins the jury award at the Festival of Cannes as the best animation film.

■ **Figure 2.191**
The short is based on keyframes which are manually drawn, then digitalized and finally interpolated with special software procedures.

1974 A.D.

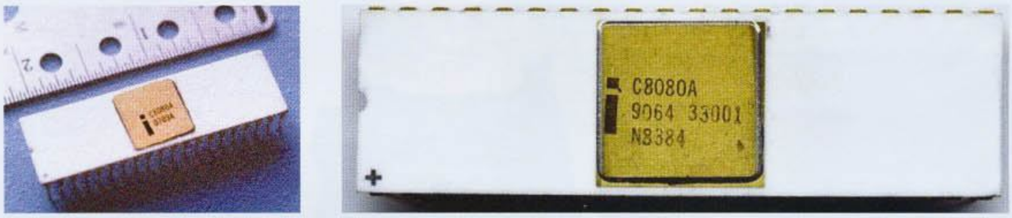
- **Edwin Catmull** invents the **Z-buffer** concept. It is a special tool able to obtain the 3D-coordinates of a pixel in space. As a result, the position compared to observer point of view is extracted too.



■ **Figure 2.192**
In a graphic card the Z-buffer is the function of automatic removal for hidden lines and surfaces. When the Z-buffering is activated every pixel gains a depth value, in addition to a color value.

1974 A.D.

- The **microprocessor Intel 8080** is born. It is equipped with a 2 MHz clock and it contains a kernel of 75 instructions. The processor has 6.000 transistors, and is able to address 64 Kb of physical memory. It is sold for $360. The 8080 is used in many historical computers, such as the MITS Altair 8800 and the IMSAI 8080. Shortly after the launching of the 8080 model, its rival, Motorola 6800, is to be introduced, and later the MOS Technology 6502, a clone of the 6800.



■ **Figure 2.193**
The Intel 8080 processor.

1974 A.D.

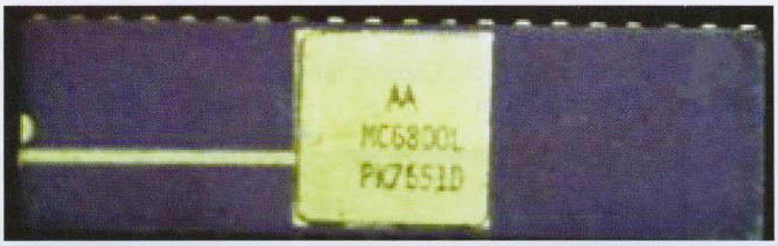
- **Jonathan Titus**, working in the Blackburg company, Virginia, develops **MARK-8**, built on the basis of the Intel 8008 processor. It is just a kit, it is not provided with power supply, screen, keyboard or chassis, but the magazine issues explain how to build them on one's own. Eight switches allow the machine's programming, one for each bit. Certainly programming this way is not that fruitful, but many computer fans are appearing in America: future hackers, or "tweakers" just can't wait. They are usually penniless chaps, full of interest and passion for electronics. The idea of having a computer of their own would be a dream come true.

■ **Figure 2.194**
Radio Electronics July 1974 cover, where MARK-8 appeared for the first time.

1974 A.D.

- **MOTOROLA** also presents its own CPU: the **6800**. A good 8 bit processor which will be used in many factory devices. The chip was designed by Chuck Peddle and Charlie Melear.

■ **Figure 2.195**
The Motorola 6800 CPU.



1974 A.D.

- The first edition of **SIGGRAPH** (**S**hort for Special **I**nterest **G**roup in **Graph**ics) takes place. It is an annual conference concerning the subject of Computer Graphics and everything related.

■ **Figure 2.196**
The SIGGRAPH logo in 2005.

SIGGRAPH2005

1975 A.D.

- The first Personal Computer, or at least the one to become famous as such, the **Altair 8800** built by the MITS company (**M**icro **I**nstrumentation **T**elemetry **S**ystem) appears on the cover of the magazine Popular Electronics in January 1975. Inputs are entered thanks to several switches located on the front panel and outputs are displayed thanks to a line of LEDs. The computer in kit version costs $395, while the assembled version costs $495. The computer's designers did not expect the success it had, thinking they would only have sold just a few hundred specimens. Ed Roberts, MITS' owner, asks his daughter to choose the name of the computer: Altair, the name of a star, used in the famous movie "Star Trek". Altair offers 256 bytes of memory (yes, you read correctly!) and it is lacking a screen and keyboard. There is a particular event appearing on the Altair 8800 announcement, which was published on Popular Electronics. It will change the life of Paul Allen and Bill Gates, persuading them that the time has come to create a new software for the new microcomputer. Altair's production will stop in 1978.

■ **Figure 2.197**
The Altair 8800 and its inner components.

1975 A.D.

- **Betamax**, developed by **Sony**, has been the first magnetic videotape recording system designed for household use. It appeared before the VHS, which is considered "technically inferior" by many, but thanks to several factors became widespread. It is said that Sony's marketing mistakes brought this bright technology to failure, by not giving user rights to other companies too, as JVC did with VHS.

■ **Figure 2.198**
In Autumn 2002 Sony states it will produce 2.000 more Betamax videotape recorders within the year, then it will stop the project permanently.

1975 A.D.

- **Benoit Mandelbrot**, working for IBM, introduces the idea of **fractal**, later used in many applications of Computer Graphics, like for example the procedural maps of 3D software. The term "fractal" derives from the Latin term "**fractus**" (broken, shattered), as the term fraction; in fact fractal images are today considered by mathematicians as objects having fractional dimension.

"It is believed that fractals, in some way, have some correspondence with the structure of the human mind and it is for this reason that people find them so familiar. This familiarity is still a mystery, and the more one deepens the subject, the more the mystery grows." Benoit Mandelbrot.

■ **Figure 2.199**
- Mandelbrot set is one of his most famous fractals.
- Benoit Mandelbrot.

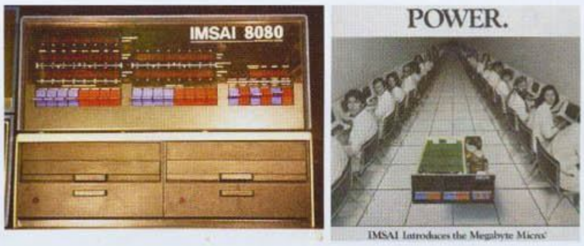
1975 A.D.

- **Martin Newell**, working at the Utah University, develops a three-dimensional model of a **teapot**, the same model that we associate nowadays with the concept of Computer Graphics. For his experiments, Martin needs a simple mathematic model, but not too simple. Thus he refers to the famous tea-pot.

■ **Figure 2.200**
The original tea-pot which inspired him is now kept in the Computer History Museum in Mountain View, California.

1975 A.D.

- Announced in summer 1975 and produced from 1976 onwards: the **IMSAI 8080**. Price: $621, exactly the same as the Altair. CPU: S-100 card (IMSAI 8080A card installed). Operating system: CP/M. A clone of the Altair 8080, better designed, equipped with a more powerful power-pack, 22 slots and a big front panel. It will be presented in June 1977.



■ **Figure 2.201**
The IMSAI 8080 and its advertising poster.

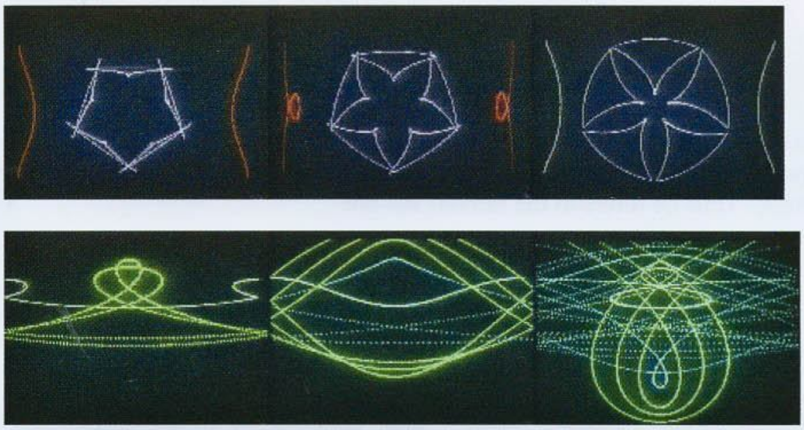
1975 A.D.

- Whilst many amateurs in the USA were trying to put together some systems as a hobby, gathering components in electronic stores, the Micro Instrumentation and Telemetry Systems (MITS) develops the **Altair 8800**, the first personal computer. Paul Allen, Honeywell's employee, was on his way to visit his friend **Bill Gates**, a Harvard student, when he saw at a newsagent the journal "Popular Electronics", showing a picture of a new computer named Altair 8800. This little grey box full of LEDs and switches will change his life and Bill's too. The magazine's title stated: "The first microcomputer in the world able to compete with commercial models". Paul Allen bought the journal and went to Bill. Both of them understood their moment had arrived. They knew the home computer market was ready to explode and somebody would have needed software for these new machines. A few days later Gates called the MITS (Micro Instrumentation and Telemetry Systems), which was producing the Altair 8800 and told them that he had developed, together with Allen, a **BASIC** language which could be run on Altair. It was a lie. They had never either written a line of code, nor did they have an Altair or the chip used on that computer. But the MITS Company, not knowing this, showed interest in seeing their BASIC. So Gates and Allen started to work hard to write the code they had promised. Gates worked on the software mostly, while Allen was trying to simulate an Altair using the school's PDP-10. Eight weeks later the two of them were sure the program was ready. Allen flew to MITS to demonstrate their creation. The day after he installed the software on the first Altair he had ever seen. If the simulation had failed the demonstration would have been a total flop. But BASIC worked perfectly from the very first moment. February: Gates and Allen accomplish the creation of BASIC and grant the license to use it to their first customer: the MITS in Albuquerque, New Mexico. BASIC is the first language written for the PC. Going back in time doc. John G. Kemeny and Thomas Kurtz had developed a BASIC language in Dart Mouth in 1964. BASIC is short for "Beginner's All-Purpose Symbolic Instruction Code". Their aim: creating a simplified language for computers to teach to students how to program. Gates and Allen understood that the compact design of BASIC would make it suitable for limitations of the first pc, which had very little memory and low computation power. MITS employs Paul Allen as software director. Bill Gates, writing to Paul Allen, uses the name **Micro-soft** referring to their collaboration and they found the Company. This is the first mention of the name Microsoft.

1975 A.D.

- **John Whitney** produces an abstract animation entitled **Arabesque**. This is a chapter in history preceding computer animation, still open to new and peculiar forms of graphic representation on computers. On the occasion of an auction of war materials, Whitney manages to buy devices for controlling anti-aircraft targeting systems, which became outdated after the Second World War. Combining some of the components together, he builds his first analogue computer with which he begins his art & research activity. Afterwards, thanks to a scholarship offered by IBM, Whitney manages to use new generation computers, which allows him to produce his masterpiece, Arabesque. It is an abstract animation representing moving geometrical shapes, accompanied by an Arab-sounding music track.

■ **Figure 2.202**
Sequential frames taken from the animation "Arabesque".

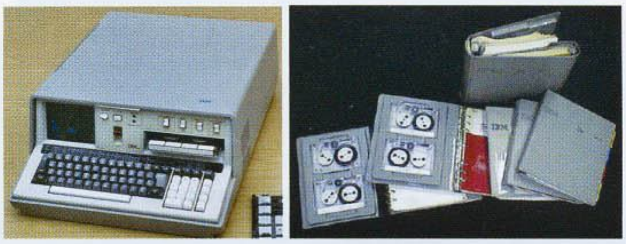


1975 A.D.

- **IBM 5100** Portable Computer: the first achievement of an IBM personal computer.

CPU: IBM owner, IC module. Original price: $20,000. Announced on the 9th of September 1975, 8 months after the Altair 8800, the **5100** can be considered as the first IBM personal computer (defined "portable" despite its generous weight). Despite this it did not capture the market because of its high price. Later on, the **5110** appeared and only in 1980 the PC **5150** did the same. The PC 5150 was called "**PC IBM**" and it conquered the market worldwide, setting a standard which was taken up in the following years. The 5110 was copied, many years later by Hewlett-Packard with its HP85 model.

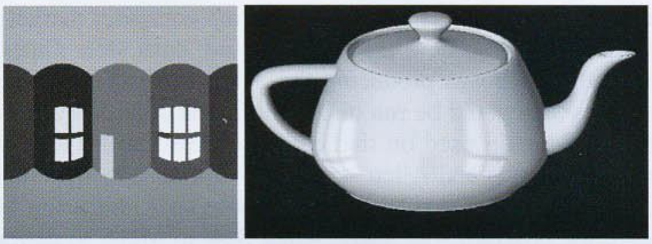
■ **Figure 2.203**
- IBM portable computer 5100.
- Handbooks and tapes included with the 5100.



1976 A.D.

- **Jim Blinn** presents a new system for generating reflections on Computer Graphics objects using environment maps.

■ **Figure 2.204**
The famous HDRI maps are later born from these experiments.



1976 A.D.

- **Ray Dolby** prepares the **Dolby Stereo system** during this year, which is to become famous in the following year thanks to the films "Star Wars" and "Close Encounters of the Third Kind". Dolby computes a code able to insert 4 channels on a stereophonic track: in addition to channels right and left, information is provided for a central channel located behind the screen, and a monophonic channel at the rear, distributed on two or more speakers. All this is obtained without the help of magnetic tracks, making the system cheap and backward compatible. Solving the continuous problem of money that weighed down factories, Dolby Stereo will become the standard for professional cinematographic sound.

■ **Figure 2.205**
Logo del sistema Dolby.

1976 A.D.

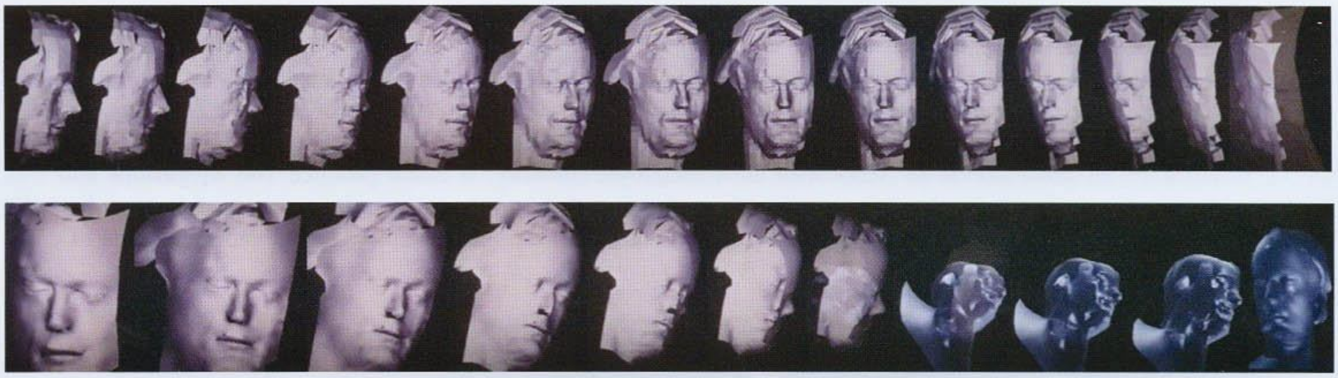
- The **AMD** (**A**dvanced **M**icro **D**evices) launches its first RAM memory on the market, the **Am9102**. Moreover, the same year they present **AMD 8080**, a varied model of the Intel 8080, which catapults the Sunnyvale Company into the world of microprocessors. In this period AMD sees an increase of its products equivalent to 60% of the yearly amount.

■ **Figure 2.206**
The first AMD processor, the 8080.

1976 A.D.

- In the film **Futureworld**, **CG** is used for the first time in 3D for the animation of a hand and a face. The boss of Peter Fonda (the main character) is digitalized. Compositing will then be used in 2D in order to materialize the whole character in front of the scene background.

■ **Figure 2.207**
- The playbill and some frames from the film.
- Some of the computer made film frames made.



1976 A.D.

- **John Cocke** works on the **IBM 801** project with the aim of developing a minicomputer with the **RISC** architecture, yet to be named so.

■ **Figure 2.208**
John Cocke and his project.

1976 A.D.

- **Mattel Electronics** introduce a hand-held game, based on LEDs and called **Missile Attack**. It is the beginning of their line of products which will include titles such as Armor Battle, Baseball, Basketball, Football and Sub Chase.

■ **Figure 2.209**
One of the first "idle leisure" games produced by Mattel.

1976 A.D.

- So far there are already at least 50 types of microprocessors available on the market! The main makes are: AMD, INTEL, Mostek, Motorola, National Semiconductor, RCA, Signetics, Teledyne Systems and Toshiba.

1976 A.D.

- In the meantime the world of mainframes continues to evolve and in this year the first supercomputer is announced, one of the most costly in history. The **Cray-1** by **Cray Research** is the first supercomputer with vector architecture. In the early 70s Cray was working for Control Data on a new machine called CDC 8600, the logical successor of the CDC 6600 and CDC7600. The 8600 was basically made up of four 7600 models placed together in a single case with a special modality of function which allowed them to work in unison with a SIMD type modality. Jim Thornton, initially Cray's colleague, during the project had launched an even more radical project, the CDC STAR-100. Contrarily to the 8600's "brute force" approach, the STAR followed a different path. The main processor of the STAR was less powerful than the one inside the 7600, but the STAR was supplied with extra hardware which drastically quickened many typical operations in a supercomputer. In 1972 the 8600 had been completed. The machine was incredibly complex and had become unmanageable. It was enough for one component to break for the whole machine to cease functioning. Cray talked to William Norris, the Control Data and CEO of the society, explaining that a new projecting was necessary. In that period the firm was facing a financial crisis and even the STAR had been a flop. Norris did not have the funds for a new machine. Cray left the firm and founded a new one, its headquarters only a few hundred meters away from the CDC, which he and other ex-CDC employees had purchased at Chippewa Falls WI. The group started looking for a new idea; initially creating a supercomputer seemed impossible: considering that even CDC had not had the funds for a new project, even more so a small, newly-founded company and practically without money. But the CTO of the society met up with some agents at Wall Street and succeeded in finding investors ready to finance the making of a new supercomputer. All that was missing was a new project. In 1975 Cray-1 was announced. The excitement due to the new machine caused a battle between the LLNL centers and Los Alamos over who would be the first client. Los Alamos won the first original model and in 1976 Cray-1 was installed with serial number 001 for a trial period of a few months. The National Centre for Atmospheric Research (NCAR) became an official client of Cray Research in July 1977, paying 8.86 million dollars (7.9 million and 1 million for the disks). The NCAR machine was dismissed in 1989. The company expected to sell around a dozen machines and instead ended up selling over eighty machines for a price between 5 and 8 million dollars each. The machines made Cray famous in his field and his company prospered, until the beginning of the 90s when the supercomputer market started to droop. Cray-1 as followed by Cray X-MP in 1982, a machine with 800 Megaflops and the first multiprocessor machine by the firm. In 1985 Cray-2 was presented, a true innovation in its area. The machine was capable of reaching 1.9 Gigaflops at peak value. This machine had a limited commercial success because of problems concerning the actual performance of the machines in the running of programs. A new model based on a rather conservative design was released in 1988, the Cray Y-MP, an evolution of Cray-1 and X-MP from which it got its name. This machine cost $700,000 and 16b models were produced in the whole world. It had a 64 bit processor at 83 MHz with an 8 Megabyte memory. It could produce 166 Megaflops and was cooled with Freon gas. The horseshoe shape come from the fact that in this way engineers were able to reduce the distance between the hundreds of components to be interconnected inside the machine. Clients had the choice of color for the chassis.

■ **Figure 2.210**
- Cray Supercomputer.
- Particular cooling system.

1976 A.D.

- **VHS** is born due to the invention of helical scanning. The Vhs, invented by JVC triumphed over Sony Betamax. Philips was defeated in its attempt to impose the Video 2000 format, the most valid of them all. VHS is the acronym for **V**ideo **H**ome **S**ystem (or also **V**ideo **H**ome **S**ervice) even though in origin it stood for **V**ertical **H**elical **S**can, based on the technique used by this system.

Already by the end of the 60s, large reel-to-reel recorders existed, produced mostly by Ampex for the television industry. Later, Sony's U-Matic was released, the first system to replace awkward and room-consuming reel machines with compact cassettes with a 3/4 inch tape, not yet apt for home use. In 1972, Philips introduced the N1500 in Great Britain, in some way an ancestor of home video recording systems, which unfortunately though suffered because of the very fact it was a pioneer with various problems connected with the short length of its cassettes; 2 or 3 were required to record a film.

1976 A.D.

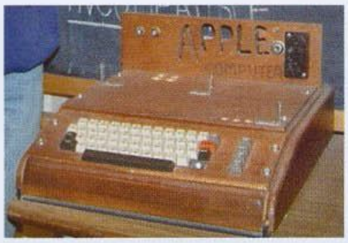
- **IBM** introduces its new computer, **System 32**, single-user, aimed at small firms. It looks like an electronic cash register machine, but is programmable in RPG (the IBM language, born at the time of the 1401 models) and has a 10Mb fixed disk.

In Italy these systems are destined to have a discreet success, due to various factors, amongst which low cost, reliability, compactness, included printer and user friendliness also regarding the programming part, thank to RPG language. The high level of interest for them is to bring many firms to demand management software developed on the basis of their specific needs. From this year onwards, the new evolution of **IBM** systems for small and medium sized firms takes place: from the **S/32** to the following **S/34** and **S/38**, and then on to the famous series of **S/36** systems, and lastly to the not less renowned family of **AS/400** systems, which have evolved to become what is now known as the **iSeries**. All these series are to be very different to each other, but have one enormous asset: they enable both developers and client firms to safeguard their computer investments, thanks to the linear ease of migrating from one series to the next and the decent compatibility of peripheral accessories.

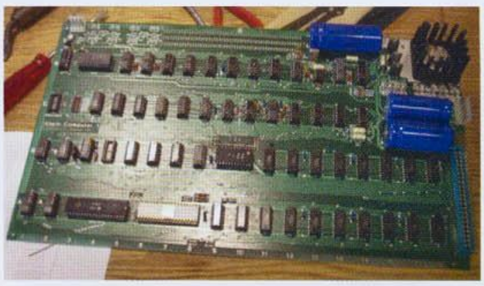
■ **Figure 2.211**
This system introduces the concept of "terminal", represented by a small screen of 6 lines each of 40 characters. Too little for an extensive use of data input, but a first step towards fulfilling programs that can guide the operator towards data insertion.

1976 A.D.

- Apple Computer is born. Outside a garage, **Jobs** and **Wozniak** build their PC with the first single-circuit motherboard, complete with video interface, 8Kb RAM and keyboard. The new PC, assembled in a wooden box, is presented to various potential investors. 200 Apple I computers are built, before the introduction of Apple II. Jobs and Wozniak continue to build their systems in their garage for another two years, before moving into what is still today their headquarters in Cupertino in California. **Apple Computer** is born.



Presentazione:	Aprile 1976	**VRAM:**	1 KB
Dismissione:	Marzo 1977	**Video:**	60.05 hz, 40 X 24 caratteri
Costo:	$ 666.66	**Audio:**	
CPU:	MOS Technology 6502	**Porte:**	
Frequenza CPU:	1 MHz	**Comunicazione:**	
FPU:		**Floppy Disk:**	
L1 Cache:		**Hard Drive:**	
L2 Cache:		**Consumo:**	
L3 Cache:		**Peso:**	
Bus:	1 MHz	**Dimensione:**	A L P cm
Slot:		**Sistema operativo:**	
RAM:	8 KB	**ROM:**	8 KB
Memoria:	Max 32 kB		



■ **Figure 2.212**
- The first Apple.
- Photo of Jobs at the exhibition of his product.
- Screenshot of the Apple.
- Jobs and Wozniak.
- Jobs' garage, where the first Apple was built.
- The first Apple logo.
- Hardware characteristics of the Apple PC.

1976 A.D.

- **Paul Allen** leaves MITS to dedicate himself completely to Microsoft. The brand **Microsoft** is registered with the secretary of state of New Mexico. The first international office is founded in Japan, on November 1st 1978, whereas in January 1979 the company moves to Albuquerque. In the photo, all the Microsoft staff in 1978. In order from top to bottom and left to right: Steve Wood, Bob Wallace, Jim Lane, Bob O'Rear, Bob Greenberg, Marc McDonald, Gordon Letwin, Bill Gates, Andrea Lewis, Marla Wood and Paul Allen.

■ **Figure 2.213**
- Microsoft staff in 1976.
- First Microsoft logo.

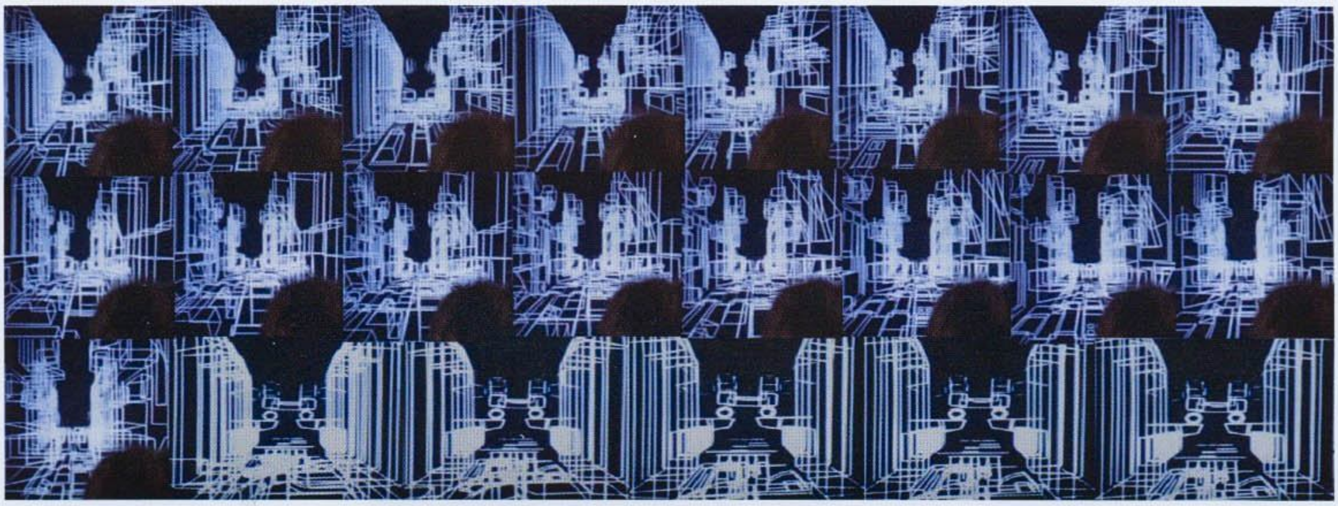
1977 A.D.

- **George Lucas** filmed **Star Wars** in 1977 and wanted to create revolutionary special effects never seen before. He then talked to Douglas Trambull, famous for his work in 2001: Space Odyssey by Stanley Kubrick. The technician turned down the American director's offer, but offered the availability of his assistant, John Dykstra, who put together a small team of university students, artists, engineers and computer experts which became the special effects department for Episode IV - A New Hope. When the following film, Episode V - The Empire strikes back, was finished, George Lucas transformed a good part of the team into **Industrial Light & Magic,** and founded its headquarters in the north of California.

■ **Figure 2.214**
- First logos of the ILM.

1977 A.D.

- In the film **Star Wars Episode IV: a new hope,** a cinematographic **vector animation** is used for the first time for the briefing sequence before the battle.



■ **Figure 2.215**
- Sequence from the film "Star Wars Episode IV - A New Hope".

1977 A.D.

- For the first time, thanks to **Frank Crow**, a practical solution against aliasing is adopted, **antialiasing** is born. It is the technique by which the parts of an image which present poor pixelization are rounded, obtaining a better visual quality. At this time Frank Crow works for the famous hardware society NVIDIA.

1977 A.D.

- Atari releases its **VCS** (**V**ideo **C**omputer **S**ystem). For the first time the system for loading games is based on interchangeable cartridges which will become the means by the which the vastest library ever of software for consoles will be created and will last for all of the golden era of videogames. Amongst various titles some are: Adventure, Asteroids, Combat, Joust and Space Invaders.

■ **Figure 2.216**
VCS console and screenshot taken from Space Invaders.

1977 A.D.

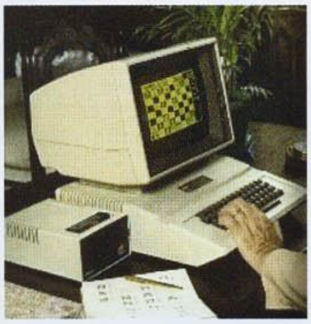
- Presented for the first time at the Consumer Electronic Show in 1976 the **Commodore PET** (acronym for Personal Electronic Transactor) is not a simple computer, but represents the efforts and sacrifices of a man who escaped from the Nazi camps to find refuge in the USA, to later give life to a repair firm for typing machines. As is possible only overseas, 25 years later he managed to buy MOS technologies, producers of the famous MOS 6502 chip: the same one to become the heart of the first computer produced by his company, the Commodore PET 2001. Unmistakable and with its own charm, enclosed in its strongly-edged metallic case with its trapezoidal shaped monitor towering at the top, the Commodore PET 2001 was immediately recognizable thanks to its non-standard keyboard, looking more like a cashier's till, and thanks to the tape reader contained inside the same cabinet. The non-standard was one of the common traits of the Commodore firm: the potential of this computer made it an immediate success (in Europe it covered 80% of the market), but the incompatibility with peripherals of other producers quickly made it obsolete. This form of computer protectionism generated a bizarre phenomenon: each Commodore computer was surrounded by necessary accessories of the same firm, creating a sort of "micro world" of spiky-looking but autonomous objects, which later appeared more rounded but nonetheless attractive. Perhaps, today, it is that tendency to row against the trend that makes the first Commodore PET and CBM so unique: similar to objects of an old science-fiction film, undisputed accomplices of the computer revolution.

Rom: **14 Mb** - Processor: **MOS 6502** - Ram: **up to 32 Kb**

■ **Figure 2.217**
Photos of the PET.

1977 A.D.

APPLE II is announced. It is a good and proper home computer with simple writing programs, calculating sheets, games and lots more.
The first computer sold with graphical functions included. Not many were produced and sold, considering the market in that period.



Presentazione:	1977
Dismissione:	1980
Costo:	$ 1295
CPU:	MOS Technology 6502
Frequenza CPU:	1 MHz
FPU:	
L1 Cache:	
L2 Cache:	
L3 Cache:	
Bus:	1 MHz
Slot:	8 proprietari
RAM:	4 KB
Memoria:	Max 36 kB, tramite slot

VRAM:	
Video:	280 X 192 6 colori, 40 X 80 4 bit
Audio:	
Porte:	Altoparlante seriale opzionale
Comunicazione:	
Floppy Disk:	opzionale
Hard Drive:	
Consumo:	
Peso:	
Dimensione:	A L P cm
Sistema operativo:	
ROM:	12 KB

■ **Figure 2.218**
The following version, Apple II plus was more successful (June '79) and even more so the Apple II version. The Apple PC only had capital letters available for writing.

1978 A.D.

- James Blinn creates the first long series of television animations called **The Mechanical Universe** and, in the same period, introduces the concept of **Bump Mapping**.

Bump mapping is a way of simulating rough surfaces, with 3D relief and 2D textures. For example: if one creates an object which represents a tree trunk, the cracks of the bark can appear three-dimensional thanks to Bump Mapping, even if they are not actually made of polygons. In this way, any changes in illumination can make these cracks appear to have depth, even though they are completely flat; thanks to Bump Mapping one can use many less polygons for creating 3D objects.

■ **Figure 2.219**
- James Blinn.
- Some frames from the series The Mechanical Universe.
- Screenshot used by Blinn for the demonstration of Bump Mapping.

1978 A.D.

- Combining their efforts, **Philips** and **MCA** develop and offer the public the **Laser Disk** format for data storage. Five years later the first CD-ROM will be presented, a natural evolution of Laser Disk. The peculiarity of Laser Disk is that it contains video data in analogical and not numerical (digital). The luminance signal is in fact recorded with a frequency modulation technique (FM); together with the video track recorded analogically it is possible to record soundtracks in digital format.

In 1982, in Japan, the first audio CD is introduced, greeted with great enthusiasm by passionate listeners of classical music. It is not until 1985 that the first read only CD-ROMs come out. In 1990 they give way to the first CD-R.

CD-ROMs have a structure similar to normal musical records: the data is organized along a spiral shaped track, a type of organization which is very different to magnetic disks. The spiral starts from the centre (unlike vinyl records) and proceeds outwards, which allows one to have CDs smaller than average (for example mini-CDs or credit card shaped ones). The spiral structure of CDs is such that it maximises the performance for sequential access penalizing direct access. One characteristic of CDs is given by its constant reading speed (CLV). This principle states that the head must read the stream of data at a constant speed, both in the outer and inner part of the disk. This is obtained by varying the speed of rotation of the disk, which passes from 500 rounds per minute in the centre to 200 on the outside, For musical CDs this is not a problem, but in CD-ROMs it is necessary to constantly change the speed of rotation, causing pauses in the reading and noises.

The physical formats of CD-ROMS are defined by documents which go by the name of Red Book, Yellow Book collectively known as the Rainbow Books. The reason for using names of colors is lost in legends: it would seem that the first version of Audio CD specifications was kept in a booklet with a red hardback cover, hence the name.

PHYSICAL FORMATS

CD Audio (Red Book)

This was the first original type of Compact Disc, put into commerce in the early '80s, conceived for the memorization of sounds.

CD ROM (Yellow Book)

Acronym for **C**ompact **D**isc **R**ead **O**nly **M**emory, these are used for the memorization of generic data. Printed with appropriate industrial machines. CD-ROMs with a large distribution have a capacity of 74 minutes/650 MB and 80 minutes/700 MB, whereas the rarer ones contain up to 90 minutes or 100 minutes/870 MB.

CD-R/CD-RW (Orange Book)

Acronym for **C**ompact **D**isc **R**ecordable. The arrival of the first CD writers made a new commercial standard necessary. The orange book actually provides three typologies:

CD-R Writable Compact Disc
MO Optical-Magnetic Discs
RW Rewritable Compact Discs

CD-i (Green Book)

Invented by Philips in 1986 and given up to Sony later on, it can be defined as the ancestor of the DVD, as it is able to contain audio, video and other integrated multimedia although with a quality comparable to that of VHS cassettes. Their diffusion is limited to the East and the USA. Also parts of these standards are the Photo CDs by Kodak.

Video-CD (White Book)

It is a format capable of memorizing audio and video data in MPEG-1 format, with a quality similar to that of a VHS cassette. It can contain up to 74 minutes of full-screen video. An entire film is usually memorized on two discs. Widely used in the East and almost unknown to us in Europe, it has been largely substituted by the DVD.

CD-XA o CD-Extra (Blue Book)

Contraction of **C**ompact **D**isc **Ex**tended **A**rchitecture. This format appeared in 1989 from a collaboration between Sony, Philips and Microsoft, and enables one to mingle audio tracks according to the Red Book, data tracks according to the Yellow Book, thus obtaining multimedia audio CDs, CD-texts, CD-plus or CD+G (Karaoke). Of the aspects of this format, fundamental for multimedia, is the interleaving technique: it is possible to physically memorize the information in a different order compared to the logical order, so that the head does not have to move around as much.

LOGICAL FORMATS

Logical formats establish the nature of the data stored on CDs and have even more bizarre and imaginative names than the physical formats.

CD-Audio

The standard used by audio CDs requires sampling of the sound waves to be carried out via the codification method called **PCM,** using a 16 bit quantization (providing 96dB of dynamics) and 44,100 samples per second (which guarantees a linearity in the spectrum of 5-20KHz). The data is memorized on the sectors of the disc, each one representing 1/75 of an audio second. It is possible to divide the data into a maximum of 99 tracks. The speed of data transmission is around 150kbit/s, also called CDx1.

ISO 9660

It is the most used format, as it uses a common base which is very simple and easily extendable.
The first level of ISO 9660 establishes only names with standard MS-DOS, in other words 8 characters per name, 3 extension ones and completely in capital letters. Every name must be different from the names of other files in the directory.
The second level, with a few compatibility issues, uses long names, up to 31 characters and shared files are not allowed.
The third level has no limitations.
In any case, as operating systems are so many and each has its own aspects, there are specific extensions for the ISO format.

APPLE-ISO (HFS)

It allows one to memorize data according to the Macintosh file system and thus allows capital letters, small letters and as long as 31 characters. Also, it memorizes other typical data of HFS file systems. This type of CD can be used natively only on Macintosh computers.

Mixed Mode

Also known as Hybrid CD. It allows one to mix, for example, a Mac CD and a Windows CD.

Photo CD

Specific Format used by Kodak. Allows one to memorize 100 images of low, medium or high resolution on a single disc.

UDF

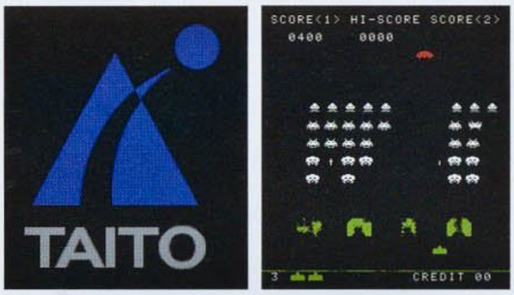
This is a new file system able to surpass the ISO 9660 limitations. Also used for certain software for packet writing CD-R/CD-RW.

■ **Figure 2.220**
Laser disk and CD-ROM compared.

1978 A.D.

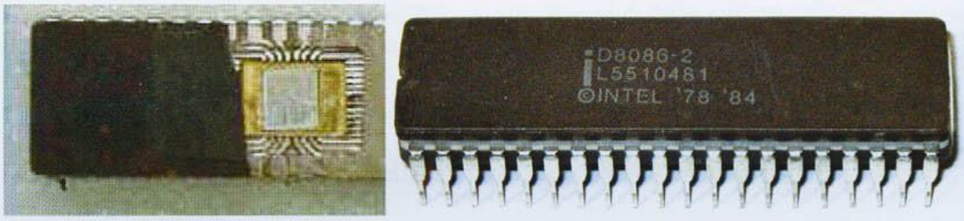
- **Space Invaders**: released to the public by **Taito** which in a short time becomes extremely popular. During its lifespan the game will generate a turnover of over 500 million dollars. The Italian license will be given in 1979 to Sidam, and the game will then be named "Invasion".

The original version of the game as supposed to have soldiers in the place of alien spaceships moving towards a sort of bunker (in fact the "terrain barriers" look very much like sand sacks used in the war and the spaceships seem to march more than fly) but the setting was changed at the last moment: Taito, in fact did not want to send out the message that killing a man could be something nice or entertaining.
Furrer Trick. This trick gets its name from its inventor, Eric Furrer, who used it to surpass his record, managing to score 1,114,000 points in 38 hours and 37 minutes. The trick is based on two simple passages: after 22 shots fired, one must wait for the Mystery Ship, destroy it and gain 300 points this way. Then one must fire another 14 and await the ship again. This goes for every level. Contrarily to what one might expect, Space Invaders came out without any copyright. Because of this, in a few months, the market was full of clones.



■ **Figure 2.221**
Space Invaders was the first electronic game to give way to the "arcade boom" and even now is one of the most cloned in history.

1978 A.D.
- **Intel** releases its **8086** processor at 16 bit. From here all the x86 architecture at the base of 90s CPUs derives. It is available in 4.77 MHz, 8 MHz and 10 MHz versions. It contains the equivalent of 29,000 transistors and is able to execute 0.33 million instructions a second. It is used in the IBM PS/2 Model 25 and Model 30. In the same month Intel releases also 8088 processors, which will be used in the first IBM PC, in the PC/XT, in the Portable PC and in the Jr PC.

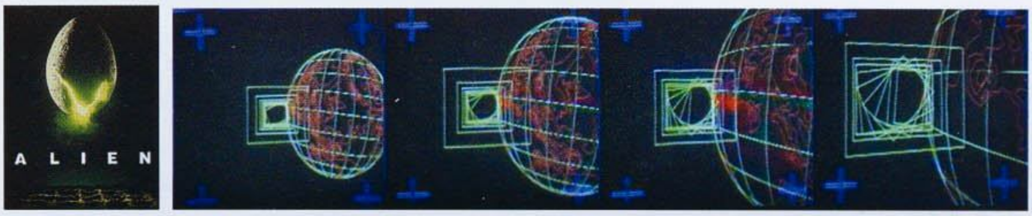


■ **Figure 2.222**
The die of the Intel 8086 and the complete chip.

1978 A.D.
- Atari presents its **Atari 800**: 8K RAM, expandable to 48, with a graphic subsystem amongst the best in that period. The price: 1,000 dollars. The design was compact and the size made the Atari 800 XL a pleasing computer to use, but the characteristics do not differ much from previous models.

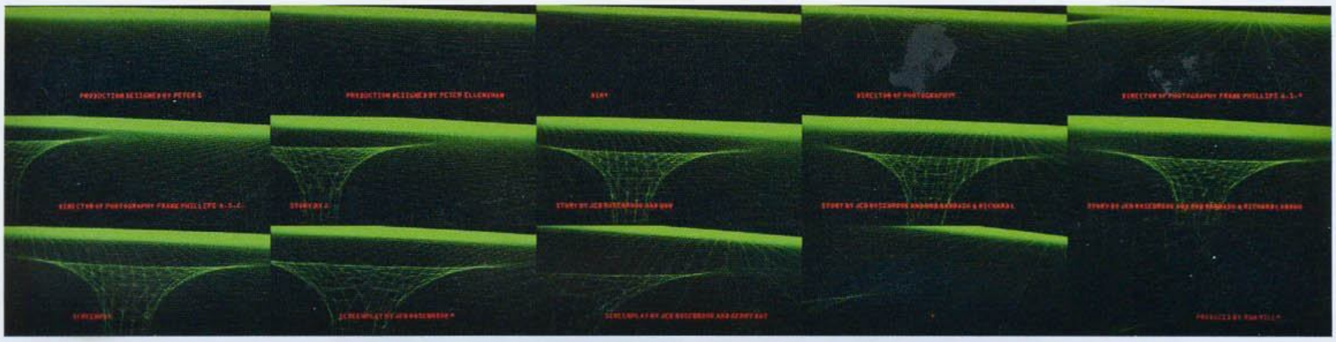
■ **Figure 2.223**
The Atari 800.

1979 A.D.
- In the film **Alien** a series of **wireframes** are used in the navigation monitor of the mother ship.



■ **Figure 2.224**
The Playbill and sequence from the film Alien.

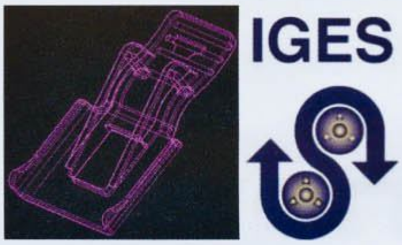
1979 A.D.
- In the film **The Black Hole**, Disney uses CGI for opening titles.



■ **Figure 2.225**
Some frames from the introductory sequence of the film "The Black Hole".

1979 A.D.

- The **IGES** (**I**nitial **G**raphics **E**xchange **S**pecification) project begins thanks to the collaboration of various CAD companies, including Boeing. It is an exchange system of information inside CAD environments, with which one can provide information regarding specific schemes, wireframe surfaces and solid representations of models.



■ **Figure 2.226**
IGES model and logo.

1979 A.D.

- **Motorola** presents their **chip 68000,** a 16 bit processor which will later support Macintosh.
The Motorola 68000 is a CISC microprocessor. It is the first member of a numerous family of microprocessors developed by Motorola and has been used by many producers of personal computers. Often the whole series is referred to as "m68k" or "68k".
Originally the MC68000 was projected to be a processor for generic use. It was, in fact, used by various producers of PCs such as Amiga, Atari and Apple. Many consoles used it. For example Sega Genesis/Megadrive, the NeoGeo and many coin-op machines. Instead in the Sega Saturn, the 68000 was dedicated to the management of sound.
The initial version of the 68000 was presented to compete with the Intel 8086, the Intel 80286 and their successors. The first version reached up to 8MHz which for those times was a very high frequency. Some instructions required only 4 clock cycles to be completed due to the efficient internal architecture. Motorola stopped making the 68000 in the year 2000, even if it continues to produce some derived products such as the CPU32. From 2001 Hitachi has continued to produce the 68000 thanks to a license granted by Motorola.

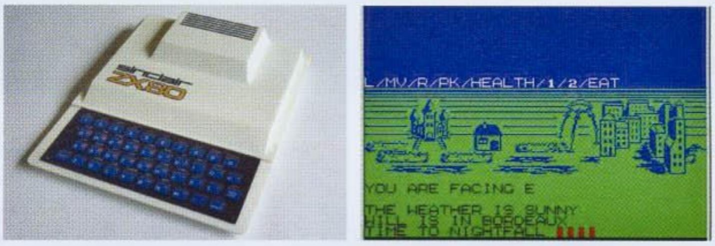
■ **Figure 2.227**
The Motorola Chip 68000.

1980 A.D.

- The history of personal computers is full of episodes which have, from time to time, undermined the security or prestige of the big producers: these events were often tied to the commercialization of small, powerful and economical computers. It is the case of **Sinclair ZX-80**, one of the most original computers ever launched.
Enclosed in a white plastic case decorated with a fake air vent, the Sinclair ZX-80 had a membrane keyboard and used the Zilog Z-80 chip with a memory of 1 Kb. It was commercialised by Clive Sinclair, an English inventor who years before founded a firm dedicated to the development of electronic appliances. The strength of the Sinclair ZX-80 was the possibility to use a simplified version of BASIC at an extremely reduced price. The construction technique of the ZX-80 allowed the Sinclair Company to rely upon a watch-making industry for the production phase, the Timex Company. This helped lower the final cost of the product: it was in fact sold only via correspondence for a price of 95 Pounds, a price which was accessible to many passionate followers of computing wishing to learn about the programming of a similar computer.

The Sinclair ZX-80 was the first computer for thousands of aspiring computer programmers and computer fiddlers, thanks also to the great literary support which it boasted: in the early '80s, in fact, the market was crammed with dozens of books regarding the Zilog Z80 and the programming of the Sinclair computer. 100,000 models of the Sinclair ZX-80 were sold. This computer was such an innovation in its area of products to represent a fundamental moment in the history of computers.

ROM: **4-8 kb** - Processor: **Zilog Z80** - RAM: **up to 16 kb** - Graphics: **64 by 44** - Color: **no** - Audio: **no**



■ **Figure 2.228**
The Sinclair ZX-80 computer and screenshot from a game.

1980 A.D.

- Created by **Tohru Iwatani** of **Namco**, it became the game which started off everything. Initially the hero is called Puck Man, but before its release in the US, the name is changed to **Pac-man**, in order to avoid vulgar name-changing.

■ **Figure 2.229**
- Namco logo.
- Puck-man playbill.
- Puck-man screenshot.

namco®

1980 A.D.

- The **PDI** is founded by **Carl Rosendahl** (**P**acific **D**ata **I**mages), which in years to come will give us masterpieces such as Shrek and Madagascar.

■ **Figure 2.230**
The founder of PDI, Carl Rosendahl.

1980 A.D.

- **Vol Libre,** a computer-generated animation made by **Loren Carpenter** from Boeing, is an amazing success at the 1980 SIGGRAPH. Carpenter draws inspiration from the fractals theory in order to create animated backgrounds to be used during flight simulations. The animation shows a sequence above a snow-covered mountainous landscape with realism unseen so far. The ILM will then use the idea to fulfill a scene in its film: Star Trek: The Wrath of Kahn.

■ **Figure 2.231**
Frame taken from the "Vol Libre" animation.

1980 A.D.

- Apple presents **Apple III**, born as an office computer. It is a commercial flop: it has very few improvements compared to the extremely popular previous version Apple II and has serious production problems, as well as hardware and software problems. The Co-founder Steve Jobs was contrary to the fans. From his point of view a computer should be a silent object.

It was not a good choice. Apple III had a definite tendency to heat up and break because of the excessive internal temperature. A famous anecdote about the Apple III tells us of a technical note advising users, in case of the machine blocking up, to lift the computer by 75 mm and drop it. The impact was supposed to reset some components of the motherboard which had a habit of seizing up.

Presentazione:	Giugno, 1980	**VRAM:**	
Dismissione:	1985	**Video:**	80 X 24 testo, 590 X 192 1 bit
Costo:	$ 7800	**Audio:**	
CPU:	SyneTek 6502A	**Porte:**	Altoparlante
Frequenza CPU:	2 MHz	**Comunicazione:**	
FPU:		**Floppy Disk:**	5.25" 143 KB
L1 Cache:		**Hard Drive:**	
L2 Cache:		**Consumo:**	
L3 Cache:		**Peso:**	
Bus:	2 MHz	**Dimensione:**	A L P cm
Slot:	4 proprietari (compatibili con Apple II)	**Sistema operativo:**	
RAM:	128 KB	**ROM:**	4 KB
Memoria:	Max 256 kB, tramite scheda		

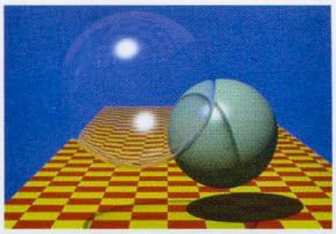
■ **Figure 2.232**
- The Apple III computer.
- Hardware features of the Apple III.

1980 A.D.

- **Bill Gates** invests 50,000 dollars in purchasing from Seattle Computer Products, a "fast and dirty" operating system, the **Q-Dos** (**Q**uick and **D**irty **O**perating **S**ystem) which will be the base for the future MS-DOS, destined to become a standard in the field of personal computers thanks to the economical power of IBM and to Bill Gates' business skills. Gates obtained an agreement for a non-exclusive license from the maker of Q-Dos, Tim Paterson, which meant that he had the faculty of re-selling the product. Later, Microsoft will close the circle by buying all of Seattle Computer Products' rights and employing Paterson himself.

1980 A.D.

- **Turner Whitted** from Bell Labs introduces a new paper regarding the concept of **ray-tracing**. To create this image 24 hours were necessary, so consequently 4 years for a 1 minute animation. A new instrument had been discovered but there were not yet powerful enough tools to use it.



■ **Figure 2.233**
- "Complex" image rendered completely in ray-trace.
- Turner Whitted.

1980 A.D.

- **Mattel** enters the market of consoles and starts with **Intellivision** accompanied by four games. The system competes directly with Atari VCS and has a decidedly better graphic look, with an audio quality without comparison. In the first year, 175,000 copies were sold with a set of 19 games each. In 1982 the number of sold consoles goes up to 2 million with a turnover of $100,000,000. In 1983 a new and innovative games peripheral comes out: Intellivoice, a vocal synthesizer to be used in certain games. In 1983 Intellivision II comes out.

■ **Figure 2.234**
Because of problems related to competition and badly chosen commercial policies at the end of 1983, the Mattel Electronics division closes.

1980 A.D.

- **Adam Osborne** launches the **Osborne 1**, first "**portable**" computer. The inverted commas are necessary because the whole thing weighs 12 kilograms. Its size, though, allows it to be stored under the seats of passengers on airlines, an innovative bonus for those times. The operating system is CP/M. The appliance costs 1,795 dollars, has a 5 inch screen (24 x 52 characters), a Z80 processor, 64 Kb RAM, and two 5,25 inch floppy drives, with 91 Kb capacity each. The computer was sold together with WordStar, Supercalc, dBase II and 2 versions of BASIC (CBASIC and MBASIC). In 1981, Osborne sells models for this "portable" for almost six million dollars; at the end of 1982, the sales are almost 69 million dollars, equivalent to almost 10,000 models each month. The Osborne Company is destined to fail two years later, on the 13th of September 1983, partly because of a blunder which brought about the premature divulgation of the features of the follow-up of the Osborne 1. Knowing that soon a better version of the PC would come out, people stopped buying the Osborne 1, preferring to wait for the new machine which never was released.

■ **Figure 2.235**
The first portable, or rather, considering size and weight, transportable PC.

1980 A.D.

- IBM introduces the **first IBM PC,** with PC DOS operating system 1.0. Based on the Intel 8088 processor, the basic version had 16 K memory (yes, sixteen), an audio cassette for memorizing programs and cost 1.565 dollars. For 3,000 dollars there was a version with 64 K memory and a 5.25 inch floppy drive which contained 160 K of data. All models had a monochromatic video card unable to visualize graphic bitmaps. There were no hard disk versions available. 13,000 copies were to be sold in the first 3 months. The IBM PC was a fundamental milestone, as it used an open architecture, allowing other producers to develop an amazing amount of accessories, to the point of creating whole computers which were "IBM compatible" and the IBM architecture becomes the standard for the following decades.

■ **Figure 2.236**
The operating system for the IBM PC is developed by Microsoft, and is called MS-DOS 1.0. It is based on CP/M, used in 8 bit computers at the end of the seventies, and on the DOS of Seattle Computer Products.

1980 A.D.

- **Xerox** presents the 8010 model, known as **Star,** the commercial successor of the Alto model. Icons accepting double-clicks, movable and overlappable windows, dialog boxes and a 1024 x 768 pixel display! As one can see in the image, the fundamental elements of a Macintosh style GUI are already present: the icons representing the various units, the document folders, the windows with scrolling bars and so on. It is the first computer in the world to be sold on the market with a GUI graphic interface with WIMP typology (**W**indows **I**cons **M**ouse and **P**ointer). Unfortunately the price was not very popular, as much as $16,595.

It was thought of as a workstation and had the possibility to be connected via Ethernet, share printers and servers, etc. The Star used a bitmap screen with windows and a mouse with two buttons, which at the time, no other computer had.

■ **Figure 2.237**
The Xerox Star anticipated in many ways what future computers would be like, in particular the Apple Lisa, Macintosh, the Commodore Amiga, the Atari ST 520 and Windows PCs as they are today.

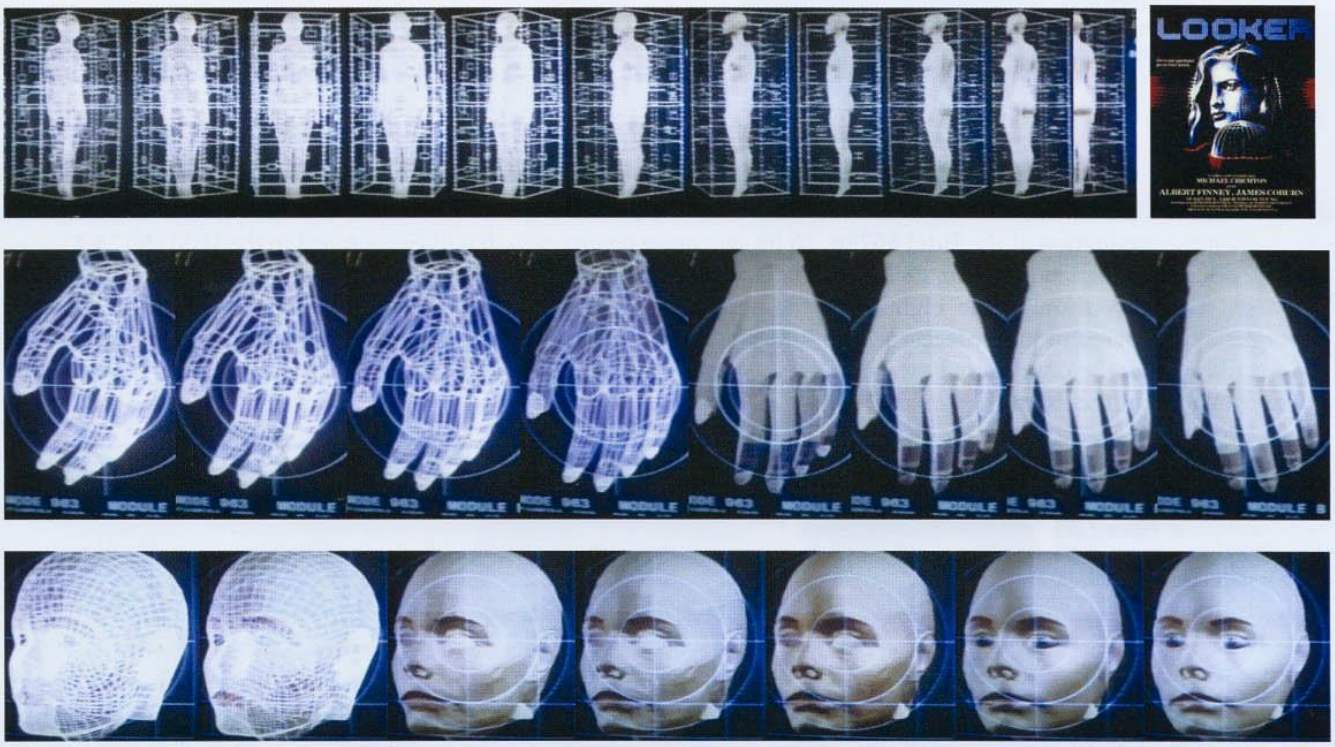
1981 A.D.

- **Nelson Marx** produces the now famous **Carla's Island**. It is a brief computer graphics film centred on a fixed framing of an island. In the sequence, the scenario transforms in an incredibly natural way. Waves form and interrupt their path breaking on the banks of the island. The light changes gracefully, reflecting on the crests of the waves with a realism never seen before. In order to animate the scene, Nelson Marx used the Cray 1 supercomputer at the Lawrence Livermore Laboratory, were he worked as a researcher. The whole scene, yet again, is based on procedural representations based on the intuition of the Benoit Mandelbrot from France.

■ **Figure 2.238**
Two frames from the film Carl's Island, made by Nelson Marx, at the LLNL and presented at the SIGGRAPH in 1981.

1981 A.D.

- In the film **Looker** for the first time a human being is used, completely illustrated in shade 3D, Susan Dey, completely in computer graphics. A plastic surgeon, Albert Finney, discovers a diabolical plan to clone models to be used for television advertisements. A Fantasy thriller without much to it, directed by the writer of "Jurassic park" and "Congo".



■ **Figure 2.239**
3 sequences taken from the film "Looker". In the film they represent the scanning procedure which the main character has to go through.

1981 A.D.

- The boom of domestic computers began with a cheap, versatile and educational model, the **Commodore VIC 20.** It was the beginning of the '80s: a new way of killing time had grown popular amongst adults and children alike. Thousands of people would sit, be it day or night, in front of this new instrument with apparently infinite possibilities. The Commodore VIC 20 was "my first computer" for a generation of people who have probably decided about their future on the basis of new prospective given by the world of computers. Based on a 6502 processor, the Commodore VIC 20 was advertised as the computer "for home, work and playtime": in effect its versatility and abundance of tools made it extremely interesting. A sound-dedicated processor, the SID, offered for the first time a new way of experimenting with music: a few program lines, explained thoroughly by "Mr VIC", the mascot of the Commodore VIC 20 manual, allowed one to simulate an infinite amount of sound effects. The possibility of instantly loading games held within cartridges seemed incredible. The graphics were rough and basic, but attractive, and, in the better looking games, made one wonder what landscapes could be hidden beyond the horizon.

ROM: **20kb** - Processor: **MOS 6502** - RAM **5.5 kb** - Text: **22x23** - Graphics: **176x184** - Colors: **24** - Audio: **4 voices**
O.S.: **BASIC**

■ **Figure 2.240**
The shape of the Commodore VIC 20 has remained impressed in our minds: two rounded plastic shells which have left an unmistakable sign in the following Commodore computers.

1981 A.D.

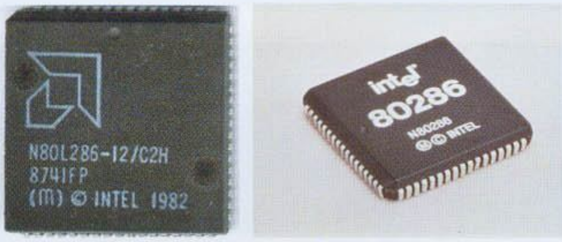
- The Lucasfilm's Computer Graphics Research Group (now Pixar) began programming their rendering engine. It was called **Reyes** (**R**enders **E**verything **Y**ou **E**ver **S**aw). Its first commercial use was in the film "Star Trek: The Wrath of Kahn". Today Renderman uses the same type of algorithm. This type of "render" had been studied to solve problems regarding the amount of memory and resources needed by a photorealistic system based on ray-tracing.
With Reyes it was possible to efficiently obtain many of the effects required in cinema productions: motion blur, depth of field, rendering of smoothed surfaces, etc... without having to rely on the complex calculations of ray-tracing.

Reyes was able to render curved surfaces which at the time were represented by parametric patches, dividing them in micropolygons each one the size of about a pixel. After, the shader assigning system would color and regulate the opacity of the micropolygons in relation to the inputs given. This type of renderer was the so-called Scanline system.

■ **Figure 2.241**
Road point to Reyes, one of the first images generated by the Reyes program.

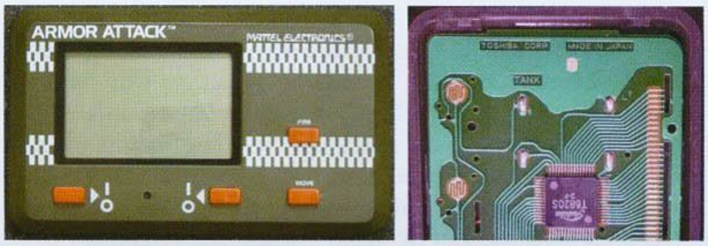
1982 A.D.
- In August the production of the **Intel 80286,** 16 bit processor begins, which will be included in the IBM PC "**AT**". Initially released in versions of 6 and 8 MHz, it later was accelerated up to 20 MHz and largely used in compatible IBM PCs from half way through the eighties to the early nineties. The 80286 processor offers twice as much clock performance compared to its predecessor, the 8086 and can exploit up to 16MB of RAM, unlike the single MB that the 8086 could use. In the computers based on DOS operating systems, this additional RAM capacity could be used only via an emulation of the extended memory.

■ **Figure 2.242**
Not many computers based on the 286 had more than 1 MB of RAM.

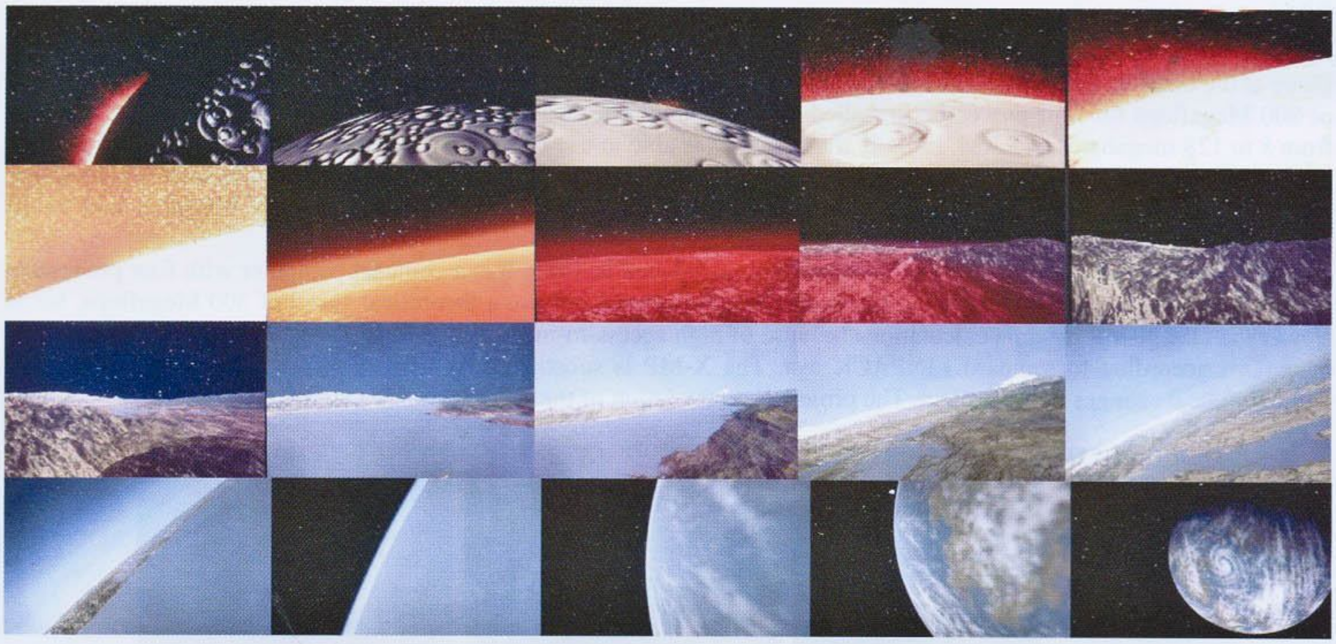


1982 A.D.
- **Mattel** produces **Armor Attack**, one of the most popular hand-held LCD games. This type of videogames is to have an enormous popularity amongst younger users because of their low price, until the new generations of consoles appear.

■ **Figure 2.243**
The portable game console and the hardware scheme, produced by Toshiba Inc.



1982 A.D.
- **ILM** produces a whole sequence from the point of view of a flying spectator for the film **Star Trek: The Wrath of Kahn**, using a system based on fractals for the terrain modelling.



■ **Figure 2.244**
Images taken from the film "Star Trek II: the Wrath of Kahn".

1982 A.D.

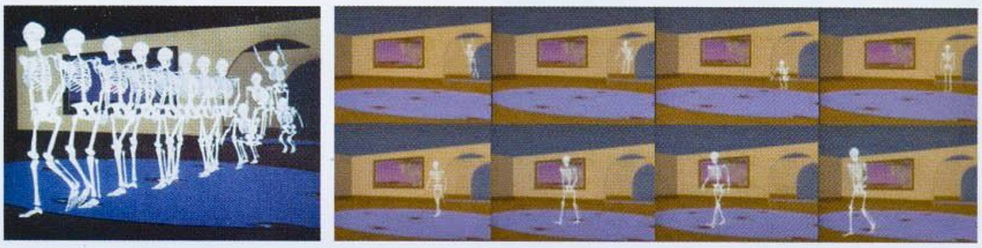
- In the famous film **Tron** CGI is used extensively. Although the film was not a great success, it catalyzed other film companies to follow the path of CG animation.

Tron is a command of BASIC, which stands for "Trace On", and searches for errors in the programming lines. The director has instead stated that he did not know about this, and that the word Tron was used deriving it from the word "electronics".

■ **Figure 2.245**
Playbill and images taken from the numerous sequences in the film "Tron".

1982 A.D.

- **Dave Zeltzer** invents **SAS** (**S**keleton **A**nimation **S**ystem). His contribution is of fundamental importance for the development of animation systems.



■ **Figure 2.246**
Animated images and sequences taken from the works of Dave Zeltzer.

1982 A.D.

- The **Cray X-MP** is produced, the fastest supercomputer in the world. The machine is developed with a horseshoe shape at the bottom. The processors are 105 MHz each, and develop a theoretical speed of 200 Megaflops per processor or 800 Megaflops for four processors installed on a machine. The X-MP is sold with one, two or four processors and from 8 to 128 megabytes of RAM. Initially the system was able to use up to 128 Megabytes thanks to the 24 bit system, but later the XMP/EA extends the level up to 16 theoretical Gigabytes. In practice the most actually used is a computer with 512 Megabytes. The XMP/EA has a theoretical peak level of 942 megaflops. In 1982 the X-MP/48 is sold for 15 million dollars plus the cost of a hard disk. 189 of them will be sold.

The successor is Cray-2, completely re-designed and introduced in 1985. It is a compact computer with four processors with 512 Megabyte up to 4 Gigabyte main memory. It is declared to have a theoretical speed of 500 Megaflops, but is slower than the X-MP in some calculations because of high access-to-memory times. In 1986 the X-MP/48 reaches 713 megaflops according to standard LINPACK test. The X-MP is substituted by Cray Y-MP which is sold from 1988 without radical changes in the project. The project is an evolution of the X-MP provided with 8 processors.

■ **Figure 2.247**
- Cray X-MP.
- One of the Main Boards of the Cray.
- Cray X-MP cooling system details.

1982 A.D.

- Rod Canion, Bill Murto and Jim Harris found **Compaq** in 1980 in Houston, Texas. One of their first products is the first portable PC in the world, or at least the first to use DOS operating system, with the simple name of **Compaq Portable**, produced in November 1982. It cost relatively little, around $2,990 and the year after, 53,000 units were sold.

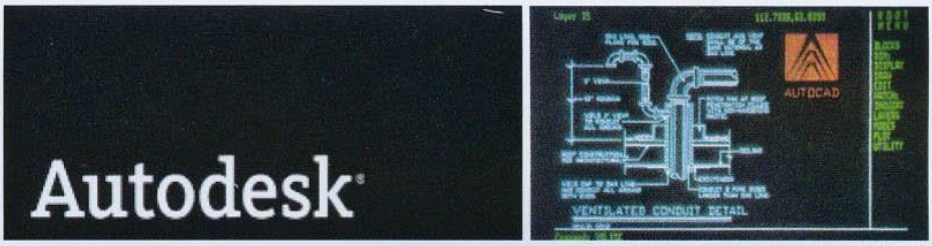
■ **Figure 1.248**
- Compaq portable in open and closed position.

COMPAQ

1982 A.D.

- MicroCAD is born which will then become the world industry giant **Autodesk.** The same year AutoCAD starts being sold.

■ **Figure 2.249**
The Autodesk logo is a screenshot from one of the first AutoCAD versions.



1982 A.D.

- **John** founds **Adobe System** with **Charles Geschke**.

■ **Figure 2.250**
John and Charles Geschke, the founders of Adobe System.

1982 A.D.

- It was September 1982. While Apple was getting all the glory for the sales of the new models of Apple II, Commodore International launched its personal computer, the **Commodore 64**. The success of the Commodore 64 was immediate: the price of the new model was half that of its competitor, but the performance was greater. What is more, the software provided seemed infinite.

In the span of a few years, the Commodore 64 entered into many a household establishing an incredible sales rate. Whoever was longing for this computer during the first half of the 80s can hardly forget the slogan shown on television: "Daddy, buy me that, so you can play too...". The Commodore 64, in fact, allowed to learn how to write simple or complex programs, or alternatively left space for creativity with the incredible amount of extraordinary games available. The programmers of the Commodore 64 exploited every single bit of memory available, creating programs which today seem impossible when considering that the processor had a speed of little more than 1 MHz and the memory was only 64 K. The Commodore 64 also had an advanced sound chip, dedicated exclusively to the management of audio effects: this is the reason why there were so many musical applications, which included the use of special keyboards to simulate a C64 piano.

ROM: **16 kb** - Processor: **MOS 6510** - RAM: **64 kb** - Text: **40 col** - Graphics: **320x200** - Colors: **16** Audio: **3 channels + noise** - OS: **BASIC** - Tape unit: **external** - **Disk: external**


■ **Figure 2.251**
With those few resources of the Commodore 64, it was possible nonetheless to synthesize a "human" voice. Anyone using this software 15 years ago will hardly forget the electronic voice sprouting from the circuits of this dream machine, as if by magic.

1982 A.D.

- A ghost wandered round the rooms of Lord Clive Sinclair's castle, a dark and powerful shadow that suddenly, in 1982 invaded the world. The **Sinclair ZX Spectrum** became one of the big rivals of the Apple and Commodore home computers, just as powerful but cheaper. Although it definitely differed to the style of its competitors, which were apparently more professional looking, the Sinclair ZX Spectrum reached the homes of millions of people. It had a small rubber keyboard encased in a black metal and plastic body. Inside the Spectrum there was a Z80 processor and a 48 Kb memory, giving life to a meticulous circuitry able to elaborate graphics with colors of 256x192 pixels an a versatile sound range.

ROM: **16 kb** - Processor: **Zilog Z80 A** - RAM: **up to 128 kb** - Graphics: **256x192** - Colors: **8** Audio: **1 channel, 10 octaves** - OS: **BASIC**


■ **Figure 2.252**
The numerous available peripherals for the Sinclair ZX Spectrum were unique and particularly original when compared to other competitors: minidisks and micro-drives, mini-printers, cartridges and other little gems from the Sinclair Company.

1983 A.D.

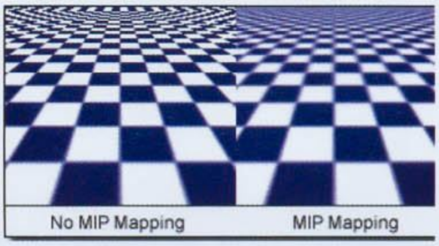
- The American franchising Radio Shack presents the **TRS-80** Model 100, **the first notebook**. It weighs 2 Kg and is an immediate success amongst journalists.

■ **Figure 2.253**
The TRS-80.

1983 A.D.

- **Lance Williams** presents a paper called "Pyramidal Parametrics" at the SIGGRAPH where, for the first time, the concept of **MIP-mapping** (**M**ultim **I**n **P**arvum **mapping**) is introduced, in other words the mapping of multiple elements which become one thing, a system to avoid aliasing in the textures.

It is a technique which allows one to partly solve the problem of the mosaic effect of the textures by building different textures to adapt to the same object the more it changes size. Imagine a tiled wall in prospective which starts of close to us and heads away into the distance. In the part near the observer, the texture is made up of many texels (individual points) which are mapped to the same number of pixels in a one-to-one proportion. As the wall draws further away, the number of tiles is always the same but the pixels available are less than the number of texels which make up the original texture, and so we have aliasing problems, as the system does not know which texel to view in that particular pixel. The MIP-mapping technique consists of calculating various copies of the same texture at different resolutions to be able to use the most apt ones according to the size of the object. During this process the uninteresting texels are filtered out as well as distortion and overlapping. In the moment the object is made bigger, the texture is changed or a new one is calculated using bilinear filtering. This technique, however, requires a lot of memory to conserve textures. Bilinear mip-mapping perfects this technique, once the appropriate texture has been chosen by carrying out a bilinear interpolation on it, in order to obtain the best color value possible for the resulting pixels. Bilinear interpolation acts upon two dimensions, horizontal and vertical, combining the values of the texels which contribute in defining the pixels and calculating an average. This can be perfected further with trilinear filtering which takes the two bitmaps closest to the desired resolution, and carries out an average on each one as in bilinear filtering, and then calculates an average of the two results. The value obtained is very accurate but work in terms of calculations is doubled.

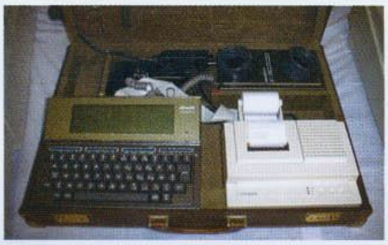


■ **Figure 2.254**
Example of textures with and without MIP-mapping applied.

1983 A.D.

- With its personal portable computer **M-10, Olivetti** once again confirms its position amongst avant-garde computer producers. The Olivetti M-10 can in fact be considered one of the first real portable computers, small and compact, energized with normal AA batteries. It is parallelepiped with a keyboard incorporated in the main body. The long liquid crystal display can be tilted for a better viewing. The microprocessor which guides the Olivetti M-10 is an Intel 8085, 8 bit with 32 Kb memory available. The operating system is Olivetti's own and contains 4 resident programs: a text editor, a small database, an agenda and a communication program. The versatility of this last software was the very thing that made Olivetti M-10 particularly practical for connection to data banks and remote computers, so much in fact that it became the favourite computer of journalists. A notable point in its favour was the fact of being able to save data and created documents on the CMOS of the M-10 (internal writable memory), without the aid of tapes or disks: these remain memorized also when the computer is turned off and can be transmitted via modem or serial port.

ROM: **32-64 kb** - Processor: **Intel 8085 C** - RAM: **up to 32 Kb** - Text: **40 col** - Graphics: **240x64** - Colors: **2**
OS: **Olivetti**



■ **Figure 2.255**
As a whole, Olivetti M-10 is an elegant computer, practical and very small (less than a sheet of A4 paper) compared to the gigantic pseudo-portable PCs of that period.

1983 A.D.

- The **Olivetti M-24** is the first Olivetti computer compatible with MS-DOS, and thus open to all available software on the market. This choice becomes necessary because of the poor success of the Olivetti M-20, the first personal computer of the Italian company, projected upon the PCOS operating system, a particular language developed by Olivetti and completely incompatible with common software.

Another major factor behind this choice was the historical alliance between Olivetti and AT&T, the great American firm in telecommunications: after the agreement Olivetti's sales of the M-24 increased incredibly, to the point that in 1985 the Olivetti Company became the second biggest producer in the World and the first in Europe.

Moreover, the Olivetti M-24 was presented with a more versatile and professional look to it: the cabinet was separate from the keyboard and could contain a disk-drive as well as a 5 Mb Winchester hard disk. The hardware was based on an Intel 8086 processor with a good 16 bits to it and better graphics than the previous model.

ROM: **8 kb** - Processor: **Intel 8086** - RAM: **up to 640 kb** - Graphics: **640x480** - Colors: **4** - OS: **MS-DOS** Audio: **Beeps** - Disc: **5 1/4" drive**

■ **Figure 2.256**
The Olivetti M-24 can be considered a milestone in the history of Italian computers.

1983 A.D.

- The **first cellular phone** is launched on the market: it is called **Motorola DynaTac 8000X**, weighing almost 800 grams, and nicknamed "the brick", because of its rather unattractive look. It was sold for 3,995 dollars: with this sort of price sales did not explode immediately. The society had foreseen that by the end of the century there would have been one million users in the world.

■ **Figure 2.257**
In actual fact, in the year 2000, in the US alone there were already 109 million users of cellular phones.

1983 A.D.

- **IBM** releases its **PC/XT**, the follow-up of the first IBM PC and also the first IBM-made with 10 Megabyte hard disk, a maximum of 640 K memory on the motherboard. It uses an Intel 8088 processor at 4.77 MHz and costs 5,000 dollars.

■ **Figure 2.258**
In 1986 the XT/286 model is installed with an Intel 80286 processor at 6 MHz.

1983 A.D.

- Microsoft releases **MS-DOS 2.0**, which introduces hierarchical directories in MS-DOS.

1983 A.D.

- **Lisa**, a revolutionary personal computer projected by **Apple Computer**, sold at the cost of 9,995 dollars. Many of the innovations in the GUI graphics in Lisa are derived from the Alto project of the Xerox PARC. The Lisa project was started in 1978, and after a long gestation, it became a project for a computer for professional use, with icons and graphic interface, which at the time was a considerable innovation.

Although many people believe that Lisa is the first personal computer to have a GUI interface and be sold on the market, this is not true. In fact the first was the Xerox Star in 1981. The first Lisa model had two 5.25 inch Floppy Disks, called "Twiggy" drives and used Lisa OS as an operating system. The processor was a Motorola 68000, it had 1 MB RAM and could use an external hard disk of 5 MB originally projected for Apple III.

Lisa was Apple's biggest commercial failure since Apple III. Potential users found Lisa to be too expensive, and relatively slow and thus were inclined to buy IBM machines or rely on other products which, albeit a slightly more complicated interface and limitations, cost much less. Lisa's final death was in 1984 when Macintosh was presented, with icon interface and a mouse. Users were not able to understand the superiority of Lisa compared to Macintosh, as multitasking and virtual memory were words without any meaning for them. The Lisa computer is a typical example of a product which was a bit too ahead of the times.

At those times 96 Kbytes of RAM memory were considered an extravagant amount and to most purposes superfluous. The generous amount provided in Lisa was seen as a waste of resources and its general low speed did not make this computer's life any easier, as a user spending 10,000 dollars for a machine expected an extremely fast machine, not a computer with an avant-garde conception but slow in everyday life practice.
Even though it was a commercial flop, Lisa received a lot of attention in any case. It was too expensive for a widespread diffusion, but for a certain period most large companies decided to have at least one ore two Lisa computers shared amongst employees. The available software for Lisa was not abundant, nevertheless when used connected to a pin printer, it allowed one to create essays and documents with page layout and graphic illustrations when instead many other computers did not have any of this. Lisa was used mostly to plan, layout and print documents. Even though many users made use of it, not many models were actually sold. These users gradually grew accustomed to icon interface and when Macintosh came out they embraced its arrival, as it provided a graphic interface at accessible prices.

■ **Figure 2.259**
- Screenshot of the graphic interface of Apple Lisa.
- Apple Lisa Model with 2 floppy disks and a hard drive.
- Apple Lisa with a single floppy disk.

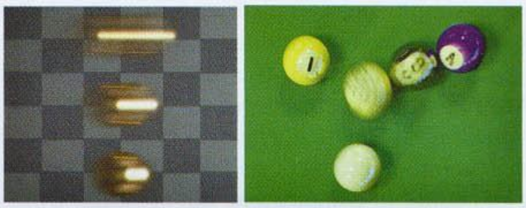
1984 A.D.

- The ILM computer animation division, which later became **Pixar**, created a famous short animation called **The Adventures of André and Wally B.** The brief animated story, which tells the story of the encounter between a bee and a lovable puppet, presents two big novelties. The forest environment scenario which the action takes place in is of an unusual quality and realism. The natural scenery, obtained with the application of sophisticated procedural modelling 3D techniques, illustrates the shapes of the trees and the thick vegetation at a level of detail never seen before. This is the background for two characters that are simply drawn but impressively natural in the way they have been animated. But most of all, the two protagonists of the story manage to communicate emotions with their convincing "body language": they move, scratch themselves and generally use typical cartoon language. But unlike their hand-drawn predecessors, they are developed in 3D by means of sophisticated computerized programs. The development of the short animation, which lasts less than a minute, put its creators in front of colossal challenges, amongst which a computational question of such an entity that it engaged the Cray X-MP/48, where 48 stands for the quantity of RAM of the supercomputer, and 10 VAX11/750.
In that same year, Lucasfilm's computer graphics division presents a new computer of its own making, the Pixar, explicitly projected for graphic visualisations. This lowered costs and time connected to the use of powerful but unspecialised calculators (like the Cray) and lays the path for what is to soon become the new direction in hardware development for the growing field of advanced graphic applications.
Albeit the more than flattering results of these public releases of the group, the relationship between the CG group and Lucasfilm becomes tricky. After The adventures of André and Wally B., the group led by Catmull feels ready to face the production of a long-length film, with which to put its new artistic direction.
Vice versa Lucas who is by now completely convinced about the potential of digital tools, wants his IL&M to continue working on digital effects, but fears that the new direction taken by the group of digital specialists is too far away from its original duties: special effects for cinema. This mutually shared conflict ends with Catmull, Carpenter and Reeves leaving the group to embark on a new an exciting adventure.

■ **Figure 2.260**
Sequence taken from the film "The Adventures of André and Wally B".

1984 A.D.

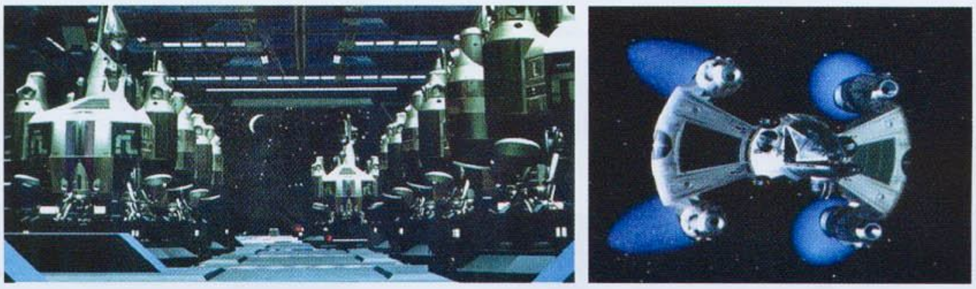
- **Lucasfilms** introduces **Motion Blur**. It is a technique used to make animations more fluid and to help perceive the idea of speed in moving objects. In order to carry out Motion Blur one must render the same 3D scene many times, changing only the position of the moving objects.



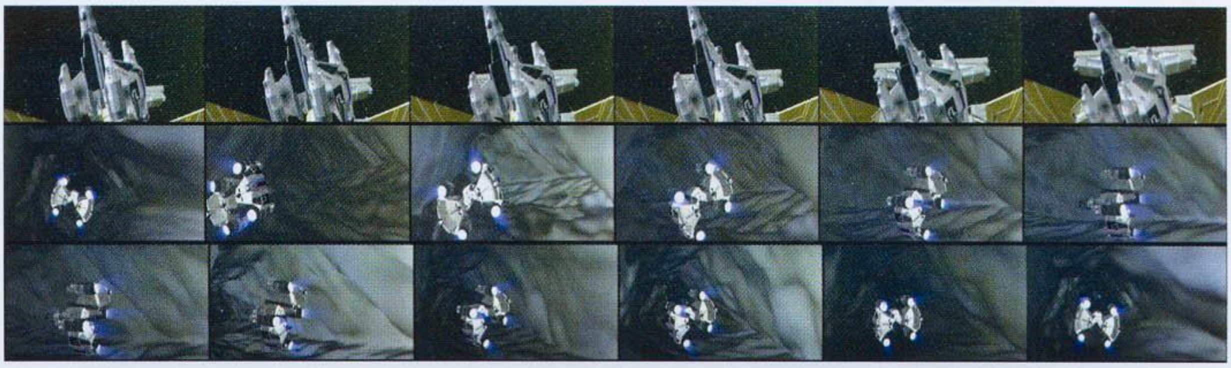
■ **Figure 2.261**
With this technique one tends to reproduce the effect given by the cinecamera's shutter.

1984 A.D.

- **The first film to largely use computerized graphics** is distributed in cinemas, The Last Starfighter. Instead of using the usual models animated with motion control, the whole space setting in the film; vehicles, planets and backgrounds are fulfilled with digital graphics. It is a heroic effort, considering it consists of creating not only the scenes, but also the software to create the scenes. A Cray super-calculator is involved in the project as well.

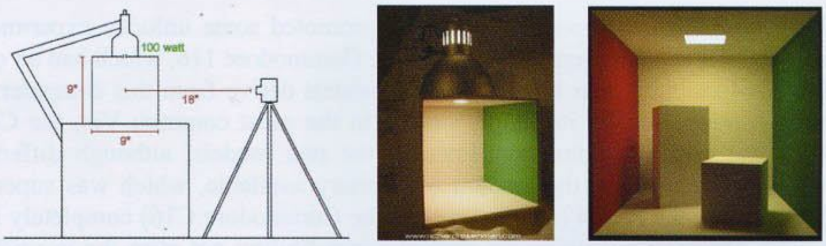


■ **Figure 2.262**
The film does not have a huge impact on the audience, but the digital scenes impress people working in the field, amongst whom Dennis Muren (maker of the "Jurassic Park" effects and part of the "Star Wars" series).



1984 A.D.

- In 1984 a group of scientists from Cornel University introduce a new algorithm which will revolutionize the method of image generation on computers. **Radiosity**, used for illuminating the scene, divides meshes into smaller portions compared to the original geometry. For each small fragment it uses its equations for the calculation of illumination, starting from the light source along to the patch, making the path of the illumination proceed until it exhausts its energy. Radiosity though, like ray-tracing, only solves the problems of a global lighting in part. In fact it does not consider effects like reflection or refraction. As ray-tracing was very efficient (in the sense that with ray-tracing reflection and refraction were perfectly reproducible), the next step was to combine ray-tracing and radiosity. The final result had to be a very robust system, able to recreate convincing and photorealistic images. Radiosity is, in itself, a system with lots of problems. For example the need to divide meshes into smaller parts is directly proportional to the final quality and to calculation power required. The more a mesh is divided, the better the GI solution is, but more resources will be necessary as well. Often artefacts can appear in particularly problematic spots, like for instance angles and obstructed areas.



■ **Figure 2.263**
The famous Cornel Box, developed in the Cornel University.

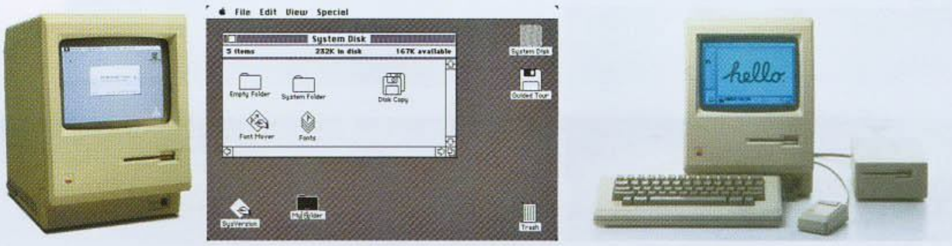
1984 A.D.

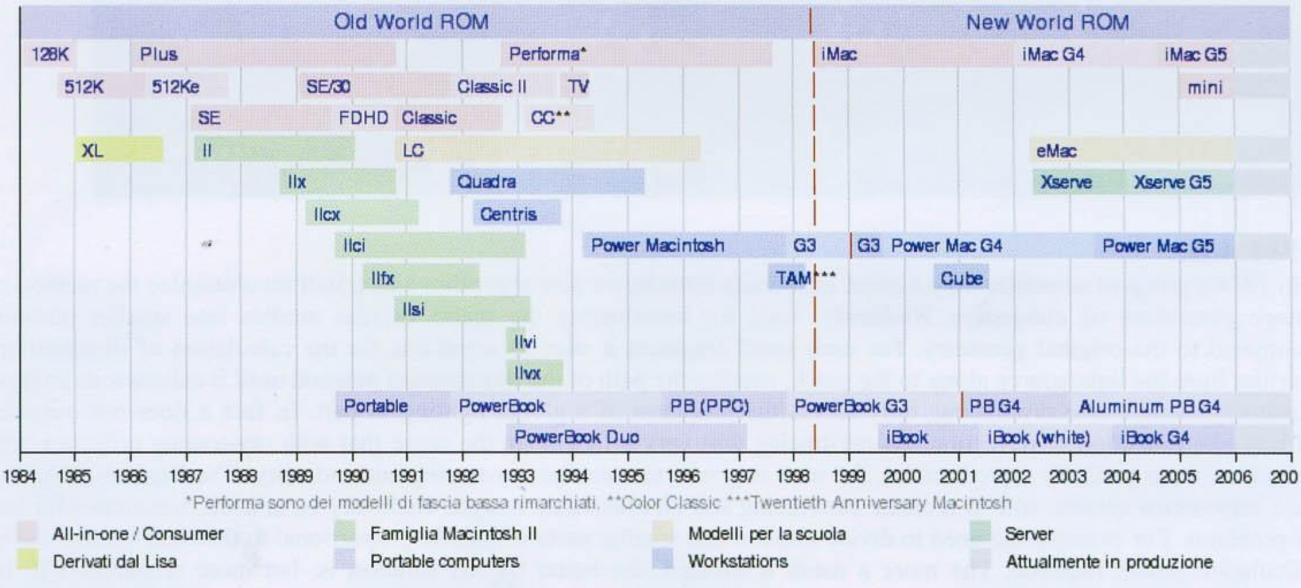
- In January 1984 the personal computer **Macintosh** is announced by **Apple** at the price of $2,500.

We are finally talking about an entirely graphic machine with an accessible price, although more expensive than an IBM PC, but certainly not as stratospherically expensive as the Lisa computer. The monitor, strictly black and white, is integrated with the CPU; the keyboard has very few keys, but efficient to the touch, as is the mouse, which only has one button. The graphic interface is simple and complete and simulates a desktop with various folders, memory devices, floppy drives, hard disk and waste-paper bin for documents to throw away, all in the form of icons. It is sold with a graphics card and the odd basic program such as a text editor and a drawing program. After the non-commercial versions of the Xerox PC Alto, this can be considered the first completely graphic personal computer open to a wide market. The success of Macintosh is undisputed: a machine which differs completely to what was and what is available. This is a particularity which Macintosh products will conserve in years to come, becoming the undisputed favourite machine of graphic workers and editorial compositors, but not only. Unlike many other personal computers, Macintosh is a closed machine. It only uses hardware made purposefully for itself, an object based operating system, and a series of programs and development languages which are completely autonomous from other computers. Even floppy disk writing is not compatible. This trait is to protect but penalize Apple for many years, until it finally sees fit to open up to the rest of the world, allowing the exchange of information and using widespread hardware, such as the Intel processors themselves, which Apple had always unwisely considered poor quality.

Initially the success of Mac was slowed by its limited amount of software. In 1985, the combination of Mac with its GUI, Adobe Pagemaker and its new laser printer by Apple, gave birth to a new low-cost solution for the publishing trade and graphic arts in advertising, an activity which would become famous under the name Desktop Publishing. The interest in Mac erupted, to the point that even today it is still a widespread reference point for typography firms, graphic arts studios and publishing firms, nowadays more bonded to the history of Apple products than to a real qualitative supremacy.

■ **Figure 2.264**
- Images and screenshots from Apple Macintosh
- Chronological chart of Apple PCs.



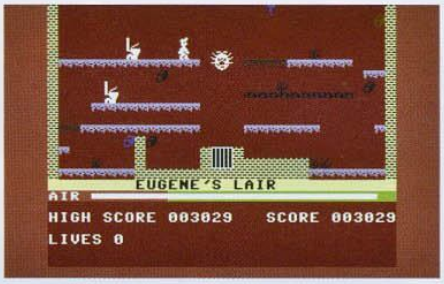


1984 A.D.

- Before following the path of the Amiga for good, Commodore promoted some unlucky experiments based on the prototype of a computer; only very few models were marketed of the Commodore 116, which had an original design but an incredibly awkward and uncomfortable rubber keyboard. Two models derive from this computer, the **C16** and the **Plus/4**. Whereas the former is characterised by its shape, similar to the most common Vic, the Commodore Plus/4 conserves the same chassis as the original prototype. However the two models, although different in shape, are practically identical: the only real difference is the amount of memory available, which was superior in the Plus/4. Nonetheless, the builders tried to make the Plus/4 (and its brother, the Commodore C16) completely incompatible with any other Commodore computer, although maintaining a mutual compatibility: not even the joysticks maintained the same scheme of the millions of joysticks for Commodore computers sold in the world.

The only positive aspect of the Commodore Plus/4 was the noticeably improved BASIC language with the addition of particular functions which made the management of graphics and sound easier: for the rest it was a complete failure.

Processor: **MOS 8501** - RAM: **up to 256 kb** - Graphics: **320x200** - Colors: **128** - Audio: **3 channels + noise** OS: **BASIC**

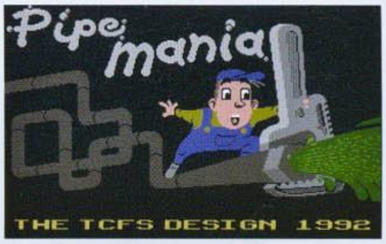


■ **Figure 2.265**
The Commodore Plus/4 and a game screenshot.

1984 A.D.

- The **Commodore 16** derives from a prototype with a completely different look to it. Its chassis is identical to that of the Commodore Vic 20 and of the Commodore 64 but is completely incompatible with them. It has a twin brother which is completely different, with a little extra memory, but software compatible.
Basically, in the C16, Commodore concentrated as many unusual aspects as it could: the result, obviously, was a failure. The prototype from which it derived was the extremely rare Commodore 116, very similar to the Plus/4 model: it maintains all the characteristics of that computer, which did not present any relevant element. A positive review of the Commodore 16 would mention the improved BASIC language, noticeably better than in previous versions, provided with special functions dedicated to graphics and sound. For everything else the Commodore 16 ended up being a failed experiment, finding itself in a moment when buyers expected ever faster and more advanced computers: assets that Commodore did not manage to instil in this unfortunate product.
It could not even boast an enviable software endowment: what made the whole product all the more bizarre was the fact that the Commodore 16 was projected to be completely incompatible with all other Commodore computers, except for the Plus/4. The latter, compared to the Commodore 16, had a greater possibility of memory expansion: if this was the only difference, why was the Commodore Plus/4 given a completely different look?

Processor: **MOS 7501** - RAM: **up to 64 kb** - Graphics: **320x200** - Colors: **16** - Audio: **2 channels** - OS: **BASIC**



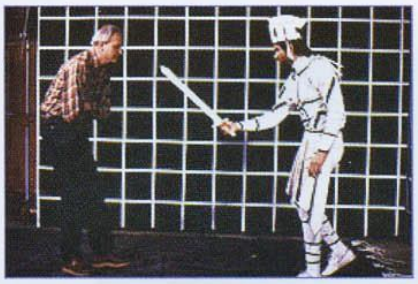
■ **Figure 2.266**
The Commodore 16 and a screenshot of a game.

1984 A.D.

- Microsoft releases **MS-DOS 3.1**: compared to version 2.0, it adds support for 1.2 MB floppy disks, a RAM virtual disc and the beginning of network support.

1985 A.D.

- In the film **Young Sherlock Holmes**, **Lucasfilms** created the first photorealistic character in the sequence of a fight between the protagonist and a knight, fulfilled in CG. The photorealistic quality of the animation was of great quality and six months work was needed for the creation of these 30 second alone. To create the effect, Pixar used the new motion blur technique and its 32 bit RGBA paint system.

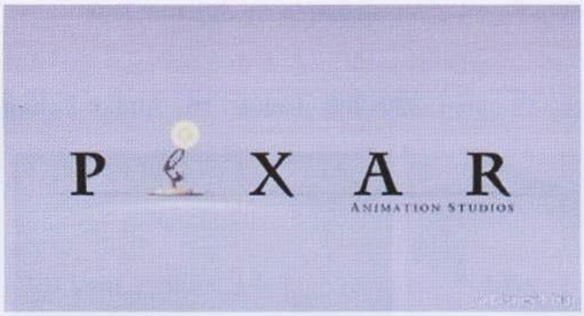


■ **Figure 2.267**
- Images of the elaboration phase in the creation of the knight.
- Images taken from "Young Sherlock Holmes".

1985 A.D.

- **Pixar** is born. Before being called Pixar it was a division of Lucasfilms (Computer Graphics Group).
It was acquired the year after by Jobs for ten million dollars, making it independent. Jobs founded the new company with Edwin Catmull, who remained a member of the executive team. John Lasseter, twice Oscar winner, film director and animator, supervises all the studio projects as executive vice-president of the Creative Department. Other members worthy of note in the executive team are Sarah McArthur (executive vice-president for production), Simon Bax (executive vice-president and treasurer) and Lois Scali (executive vice-president and general counsellor). It was specialised in the development of CG software, but with a section dedicated to hardware development. The main softwares created were REYES (Renders Everything You Ever Saw), CAPS, Marionette, animation software, Ringmaster. Amongst their most famous short-length films, some worth mentioning are Luxo Jr. (1986), Redis Dream (1987), Tim Toy (1988), KnickKnack (1989) and Geri's Game (1997). It will become famous to the public with the film Toy Story (1995).

■ **Figure 2.268**
The Pixar logo which appears in the introduction to all their films.

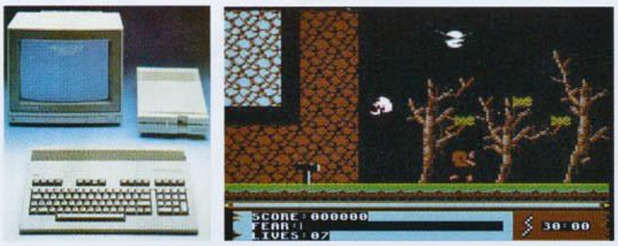


1985 A.D.

- Launched in 1985 at the Las Vegas Computer Show, the **Commodore 128** was projected with the aim of competing directly with the Apple Macintosh and the IBM PC. Although its target was the area of less professional users for Commodore products, the 128 had all the necessary assets to be a computer of discreet potential. It had a processor compatible with the glorious Commodore 64 and could be used for all the programs of its ancestor. The hardware characteristics, however, provided users with a large range of professional uses: the Commodore 128 could count on a Z80 processor which allowed the use of CP/M and of programs written in this language. Software created for Osborne and Kaypro computers could actually run on the Commodore 128. The resident BASIC had improved discreetly compared to the previous models, integrating advanced graphic functions and particular sound possibilities. The Commodore 128 was a powerful computer and dangerous enemy for rivals, but it was not a huge success.

ROM: **48 kb** - Processor: **MOS 8502** - RAM: **128 kb** - Text: **80x25** - Graphics: **640x200** - Colors: **16**
Audio: **3 channels, 6 octaves** - OS: **BASIC**

■ **Figure 2.269**
Fans forgot about it quickly "thanks" to the release of an even more advanced brother, which was born in that exact period: the Amiga.



1985 A.D.

- The idea of the Amiga remains one of the most tormented pages of computer history since the Apple I. The **Amiga 1000** was born from an idea of three computer experts who, in 1982, threw themselves into the building of the first 16 bit console for videogames, based on a Motorola 68000 processor. The Amiga 1000 project was so secret that code names were used for the chips, which conserved this fake name until final commercialization. The combination of three different processors gave the Amiga 1000 performance, unreachable even by modern computers available at the time: however the videogame crisis of the early eighties forced projectors to change path. From the initial project of a console, the project moved to become that of a personal computer which was a choice made easier by the fact that, from the very beginning, hidden expansion possibilities had been made possible. The society founded by the three colleagues found itself all the same in difficult conditions: on the verge of being bankrupt, they presented the prototype of the future Amiga 1000 at the Consumer Electronic Show. The public reaction was exceptional, but no concrete chances of fulfilling the project showed up. When the economical situation became unsustainable the three colleagues of the Amiga project decided to sell the society to an outside company: after a few months' hesitation, Atari proposed to finance the project, demanding in change the rights on the three processors used in the project. The proposal stimulated the competitive instinct of Commodore which made a surprise offer of an irrefutable amount: it was 1984 and the new-born Commodore Amiga Association celebrated its baptism with a series of lawsuits from Atari.

The agreement was settled: the first model, the **Commodore Amiga 1000**, defeated every other competitor on the market when it was released on the market. The exceptional graphics (professional computers at the time only represented tones of grey color), the sound was comparable to that of hi-fi setups and the processor speed were reached only by the other major competitor, the Atari 520ST, soon overtaken by following models of the Commodore. The era of the Commodore Amiga had just begun, spangled with difficult moments but also unforgettable success.

ROM: **8 kb** - Processor: **Motorola 68000** - RAM: **512 kb** - Text: **80x32** - Graphics: **640x512** - Colors: **64 on 4096**
Audio: **4 stereo channels** - OS: **Amiga DOS** - Disk: **3 1/2"**

■ **Figure 2.270**
The new computer based on the Motorola 68000 chip at 7.16 MHz was endowed with the 3 custom chips Agnus, Denise and Paula dedicated respectively to RAM and I/O management, graphics and sound.

1985 A.D.

- **IBM** introduces the **PC/AT**, a personal computer based on an **Intel 80286** processor. It is the first personal computer to use the ISA standard (**I**ndustry **S**tandard **A**rchitecture) with a 16 bit bus and 6 MHz. The PC/AT later adopts the new operating system signed by Microsoft: MS-DOS 3.0, which supports 5.25" floppy holding 1.2 MB and 20 MB hard disks and a new keyboard, similar to the one used currently. Two versions are to be released. The first, model 1, has 256Kb RAM, 2 floppy disks and a color monitor, whereas model 2 has 512 Kb RAM, one floppy disk and the same monitor and colors of the previous version.

■ **Figure 2.271**
The IBM PC/AT.

1985 A.D.

- The **Cray-2** is completely redesigned and introduced in the market. It is a compact computer with four processors of 250 MHz, each with a range from 512 Mb to 4 GB of main memory. It claims a theoretical speed of 1.7 Gigaflops and each processor generates a power of 488 Megaflops.

■ **Figure 2.272**
- The complex internal structure of Cray-2.
- Two installations of Cray 2.

1985 A.D.

- In October **Intel** announces its 32 bit chip called **80386** with memory management. It is able to utilize 4 GB of RAM. It contains the equivalent of 275,000 transistors and is able to carry out 6 million instructions a second. The commercial success of the 80386 brought Intel to develop different versions of this processor in order to cover more areas of the market: to avoid confusion with the basic version, it was re-named 80386DX. In 1988 the 80386SX was introduced, a low cost version of the original core.

■ **Figure 2.273**
The Intel 80386 chip and the heart of the processor.

1985 A.D.

- Microsoft releases **MS-DOS 3.2,** which compared to the 3.1 version, adds 3.5inch floppy support at 740K, the XCOPY command and partitions up to 32 MB on hard disk.

1985 A.D.

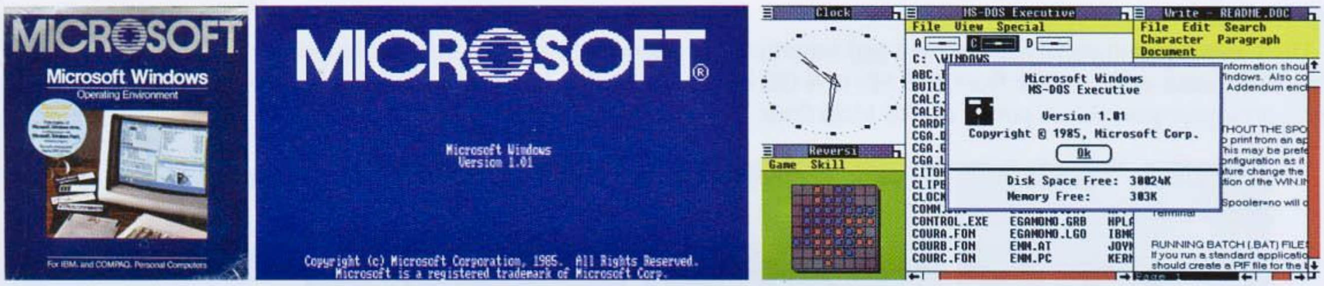
- **Microsoft** develops **Windows 1.0**, introducing typical Macintosh traits in DOS-compatible computers. The project includes a graphic interface which allows users to carry out operations selecting icons and words on the screen using a mouse.

Unlike the following versions, Windows 1.0 offered limited multitasking for existing MS-DOS applications; instead a lot of attention was placed in creating an interaction paradigm, an execution model and stable APIs which could be used in the future for native applications. Twenty years later, not only can one still run programs written for Windows 1.0 on XP, but it is possible to recompile their source code to obtain a modern application, applying only slight modifications.

Windows 1.0 is often wrongly considered "a mere interface for MS-DOS", a consideration often applied to following versions as well. It is true that Windows 1.0 was launched from MS-DOS and could itself recall MS-DOS functions and interface programs were executed by .exe files just like MS-DOS programs. However, Windows .exe files had a "new executable" format which was theirs only, which Windows only could elaborate and which, for example, allowed one to load parts of code and data upon request. Applications could administer the amount of memory to be used only through the Windows memory management system which would implement a scheme for software virtual memory and allowed the execution of programs greater than the available RAM amount.

Considering Windows 1.0 as an "interface for MS-DOS" means forgetting that the system was projected with the aim of being a graphic environment for applications and not an actual operating system. Windows 1.0 contained original drivers for video cards, mouse, keyboards, printers and serial ports and applications could only recall APIs based on these drivers. Considering that at the time the graphic possibilities of MS-DOS were very limited and given the limited utility of other services that it offered, MS-DOS applications had to access hardware directly (or sometimes exploit the BIOS) in order to do their work. So, rather than being an interface for MS-DOS, Windows 1.0 was a completion for it, and partly ended up substituting it.

The system requirements for Windows 1.0 were MS-DOS 2.0, 256 KB RAM, double faced disk or hard disk. The first version of Windows would run the shell, known as "MS-DOS Executive". Other programs were Calculator, Calendar, Scheduler, Clipboard viewer, Clock, Control Panel, Notebook, Microsoft Paint, Reversi, Terminal, Write and the command prompt. One of the interesting aspects of the system was the fact that windows could not be overlapped but only placed next to each other (tiled). Only dialogue windows could appear in front of other windows.



■ **Figure 2.274**
Packaging, opening logo and graphic interface of Windows 1.0.

1985 A.D.

- **Steve Jobs abandons Apple** because, in his opinion, the company has lost its direction. Jobs finds it hard to stand the presidency of John Sculley. The administration counsel of the company decides to remove Jobs from directing the society that he himself founded. He remains in Apple until September of that year, then he resigns. In actual fact Jobs already has a good idea of what his plans are. He wishes to re-invent the world of computers again, revolutionizing personal computers yet again, after already having been the author, alongside Wozniak and Apple, of the calculator revolution, a tool for all. He wants to create the most powerful, elegant and user friendly machine ever. With this in mind, he founds **NexT Inc**.

At first, in order to create his new firm, Jobs sells almost all of his Apple shares, making 7 million dollars. Obviously these are not enough. Also because a computer needs an operating system to run on. Mac OS was unthinkable and certainly not MS-DOS or Windows either. In his roaming from one university to another, Jobs reaches the Carnegie Mellon University and encounters a young student working at the kernel Mach 3. Jobs is impressed and decides to employ the young and talented man as vice head of software for the NexT society. His name is Avie Tevanian and currently he is the vice head of Apple operating system development. Soon after the two start working on NexTstep, the new operating system, and, with the collaboration of Adobe, the computer Nextcube.

On the 12th October 1988, in San Francisco, Steve Jobs organizes a public meeting to present the result of his work. In front of an enthusiastic crowd he starts off with the phrase "it's great to be back!" and then proceeds to illustrate and present, for the following 45 minutes, both Nextcube and the integrated operating system, the NextSTEP (version 0.8). Version 1.0 will be available 1 year later for developers only in the US. It is triumphantly greeted. The Nextcube is a great machine, powerful and easy to use. It holds a Motorola 68030 processor which is then replaced with a 68040, at 25 MHz (at the time Macs were running at 20Hz). The screen is initially black and white, but it is possible to connect a color monitor thanks to the Nextdimension card, equipped with a RISC processor. Amongst its characteristics: very silent ventilation, Ethernet and TCP/IP standard, screen and printer managed directly by Postscript language. It has 8Mb RAM extendable up to 16, a screen resolution (17 inch monochromatic monitor with integrated microphone and loudspeakers) of 1120x832 pixels. The energy feeding of the machine adapts to voltage levels of all countries in the world. The SCSI hard disk, from 330 to 660 MB and the CD reader are optional. For the storage of data there is a magneto-optical drive of 256 MB, rewritable. The machine is virtuous inside as well as outside. An operating system which is decidedly ahead of the times. It is derived from UNIX, and has some characteristics that Mac users know like the palm of their hands. It is multi-user, multitasking and multiprocessor, but this last one is an option. It is able to recognize both Mac and PC format floppy disks. The graphic interface it boasts has no equals, not only when compared to PC competition, but also compared to Mac. Dock is also a new appearance (now a part of Mac OS X), and the management of folders and applications is handled by Workspace Manager. Application menus are vertical, so as to take up less space. If an application does not respond it is possible to close it, forcing the exit. But the Achilles ' heel of this product is already clear, and will bring this adventure to halt. It is too expensive: $6,500.

San Francisco will be the chosen stage for the launching of the new machines: another Nextcube, but the novelty is the Nextstation. This new computer has a Motorola 68040 processor at 25 MHz, 33 MHz for the Color version. It has 8Mb of available RAM expandable up to 128, a hard disk of 105 MB or 400 MB, floppy disk drive, SCSI-2 port, Ethernet, 2 RS423 serial ports.

In the meantime Apple is searching for a new operating system, precisely since 1987. There are many ideas, the problem in fact is that there are too many which lead nowhere, and cause the company to lose money, market shares, users and time. The Copland project ends completely, and Apple looks for a new idea to finally get out of its rut. There is talk about using Solaris by Sun, property of Windows, until instead the new idea becomes BeOS. This system was produced by a firm founded by Jean Louis Gassée, founder of Apple France and subsequently prominent member of the bitten apple company, after the exile of Jobs. From Cupertino, Gassée resigns and founds BeOS. Contacted by Amelio, he illustrates a few possibilities about how to implement his operating system on Apple computers and everything seems to go in the right direction. Amelio himself confirms the intention to purchase the whole company, providing complete inclusion of its employees within Apple. Jobs reappears and calls Amelio, and together they discuss the future prospects of Apple. Jobs strongly advises against the use of BeOS, and at the end of the day, is not far from the truth. Not many important applications ran on BeOS, and many years would have been necessary to develop this possibility. In November 1996, negotiations which lead to Apple buying NexT, for 400 million dollars and which will see the return of Jobs into the company he founded.

■ **Figure 2.275**
The Cube, ancestor of the Apple Cube, and the Nextstation, presented in 1990 in San Francisco. Graphic interface of the operating system.

1985 A.D.
- In Italy the new **RTMS** (**R**adio **T**elephone **M**obile **S**ystem) starts functioning in Rome and Milan.

1985 A.D.
- Pixar produces the **PIC** (**P**ixar **I**mage **C**omputer), with the intention of producing a machine capable of powerful performance in visualization and simulation.

■ **Figure 2.276**
Together with Disney, Pixar developed CAPS (Computer Animated Production System) which enabled images to be colored by hand. Pixar was then purchased by Steve Jobs in 1986.

1986 A.D.
- **Softimage** is founded by Daniel Langlois in Montreal, Canada. In the following years it works in the field of production and 3D animation, 2D cell animation, compositing and visual effects software development.

■ **Figure 2.277**
Softimage is bought by Microsoft in 1994 and then in 1998 by Avid Technology.

1986 A.D.
- **Mental Image** is founded in Berlin.

1986 A.D.
- **The first ARPAnett node and access in Italy** is activated at the CNR in Pisa, thanks to the work of the researcher Blasco Bonito, via the Atlantic satellite network called SATNET.

■ **Figure 2.278**
Blasco Bonito.

1986 A.D.
- Two years after the launching of the first 16 bit Commodore, the **Amiga 500** is marketed, targeting a wide group of potential buyers, and created with the aim of permanently defeating its toughest rival: the Atari 520ST. Based on the same project as the Commodore Amiga 1000, it had twice as much memory and had the operating system directly integrated in the ROM, reducing production costs thanks also to its compact configuration.
The Commodore Amiga 500 was one of the greatest commercial successes in the field of computing: defined as "the best electronic consumer product in 1987", in its four years of life on the market it was purchased by three and a half million people, reserving itself a place in computer history. The feature of the Amiga 500 that most attracted the public was the concentration of the most sophisticated computer resources available in one single machine. Moreover at a reasonable price and with good expansion possibilities. The availability of software for the Commodore Amiga 500 was incredible: from graphics to sound, from desktop publishing to videogames, the Amiga seemed to the eyes of everyone to be the "ultimate" computer. The Atari competitor had been beaten, losing its priority in multimedia software development for home computers. The Commodore Amiga 500 left its mark on a whole generation of people, both young and not, forced to pass long and unforgettable hours in front of a monitor ablaze with colors and sounds.

ROM: **512kb** - Processor: **Motorola 68000** - RAM: **512 kb** - Text: **80x32** - Graphics: **640x512** - Colors: **64 on 512**
Audio: **4 stereo channels** - OS: **Amiga DOS** - Disk: **3 1/2"**

■ Figure 2.279
The famous Amiga 500 and a screenshot from the game Indianapolis.

1986 A.D.

- Released contemporarily to the Amiga 500, the Amiga 2000 had the same hardware and software characteristics. It was however intended for professional use, by integrating expansions common to IBM PCs. In the sturdy cabinet of the Amiga 2000, as well as the room for three disk drives and two hard disks, there was a motherboard able to host up to four supplementary cards, destined for various uses: it was thus the perfect computer for professionals, amateurs and passionate "tweakers".

The Amiga 2000, could also emulate a computer compatible with IBM systems, thanks to a few costly but efficient cards. The advantage for the final user was that of being able to reunite, in one product, all the multimedia possibilities of the Amiga 2000, and the typical expandability of a professional product: the high price pushed the target market to a higher level, but guaranteed a certain level of popularity. The expandability kept it selling for five years until 1992, and then in that year production ceased.

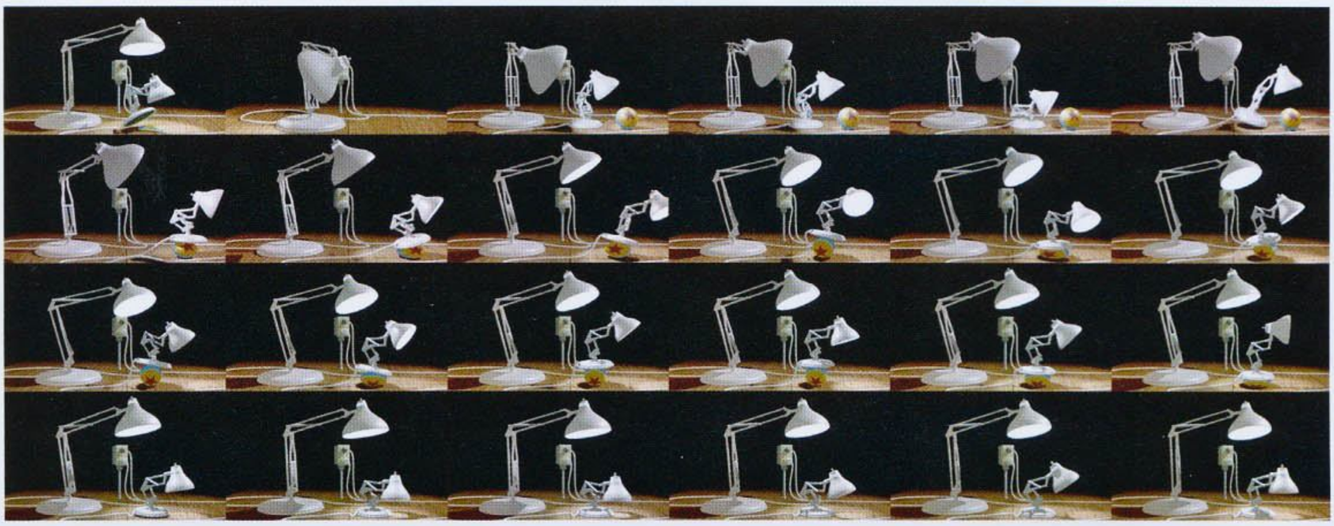
ROM: **512kb** - Processor: **Motorola 68000** - RAM: **1 Mb** - Text: **80x32** - Graphics: **640x512** - Colors: **32 on 4096**
Audio: **4 stereo channels** - OS: **Amiga DOS** - Disk: **3 1/2" built in**

■ Figure 2.280
Its considerable capacity in graphic activities brought it closer to the world of television and cinema: the Amiga 2000 was used for the creation of special effects and three-dimensional simulations.

1986 A.D.

- Pixar enters history with its **Luxo Jr**. For the first time a CGI-based film is nominated for the Academy Award. The two main characters, father and son, are strongly personalised. They are not humans or animals though, but two mechanical objects, two desk-lamps to be precise. Little Jr Luxo, his father watching him with a pleased expression, explores the world of toys, playing with a small ball. With great enthusiasm he plays, jumps, rolls, bounces his new present until, to his great disappointment, it explodes and deflates. But a new ball is ready to enter the scene.

Thanks to the fluidity, the natural movements and the psychological subtleties that the Pixar animators are able to put into the articulated lamp-protagonists, the audience immediately finds itself on the same wavelength as the two characters, and forgets that they are mechanical objects.



■ Figure 2.281
A series of images of the main scene of this short-length film. The child-lamp jumps on the ball, accidentally piercing it. A few seconds later a new one will appear and he will begin playing again, even happier still.

1986 d.c

- **Kajiya** gathers the discussions on Global Illumination written up until then and in 1986 writes what is to become the foundation of current rendering engines: the **Rendering Equation**. Kajiya applies the Monte Carlo method to solve this equation and suggests new techniques to accelerate these calculations.

Many development environments have the aim of creating photorealistic images, undistinguishable from real photos. The implementation of realistic renderers always has, at the basis, the simulation of physics, at the heart of the process. The term **based on physics** indicates the use of models and approximations which are very generic and widespread outside the rendering environment. A series of techniques has gradually become common practice amongst graphic artists. The basic concept is quite easy to grasp, but not treatable by merely computational means; there is not one single and elegant algorithm, spread amongst commonly used rendering programs. In order to meet the demand for robustness, accuracy and practicality, each implementation uses a mixture of techniques differently.

■ **Figure 2.282**
Kajiya and his famous formula.

$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_{\Omega} f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}'\cdot\vec{n})d\vec{w}'$$

1986 A.D.

- The firm **Thinking Machines** releases its first pioneering super-calculator. The **Connection Machine CM-1**.

The **Connection Machine** is a series of supercomputers developed by Dann Hillis at the beginning of the eighties at the MIT, in contrast with the traditional architecture of Von Neumann. The CM-1 was a parallel computer, based on a hypercube topology which connected the various nodes. Each node was a very simple processor endowed with its own memory. The Connection Machines were initially projected for artificial intelligence applications and symbolic calculus, although they were successful in applied science which required powerful calculation power. Hillis and Sheryl Handler together with their team developed the CM-1 and the CM-2, which in some configurations could reach the amount of 65,536 processors. The single processors were very simple, processing one bit at a time. The CM-2 presented in 1987 added 3,132 floating-point coprocessors. Each coprocessor is shared by 32 processors. Two variations of the CM-2 model were built, the smaller CM-2a with 4,096 or 8,192 processors and the faster CM-200.

With the arrival in 1991 of the CM-5, Thinking Machines abandoned the hypercube architecture of the CM-2 in favour of a new MIMD architecture based on a Fat tree network which manages a series of RISC SPARC processors. The last CM-5E replaced the SPARC processors with the faster SuperSPARC ones.

The whole list of Connection Machines in chronological order is: CM-1, CM-2, CM-200, CM-5 and CM-5E. The Connection Machines are famous for their choreographic design. The CM-2 was a grey cube with various red LEDS on the surface. The CM-5 was a tower with a wide control panel full of LED diodes. During functioning, the LEDs continued flashing to show node activity. It was even used in the film "Jurassic Park", in the control room on the island.

■ **Figure 2.283**
Two installations of the Connection Machine CM-1 in on and off states.

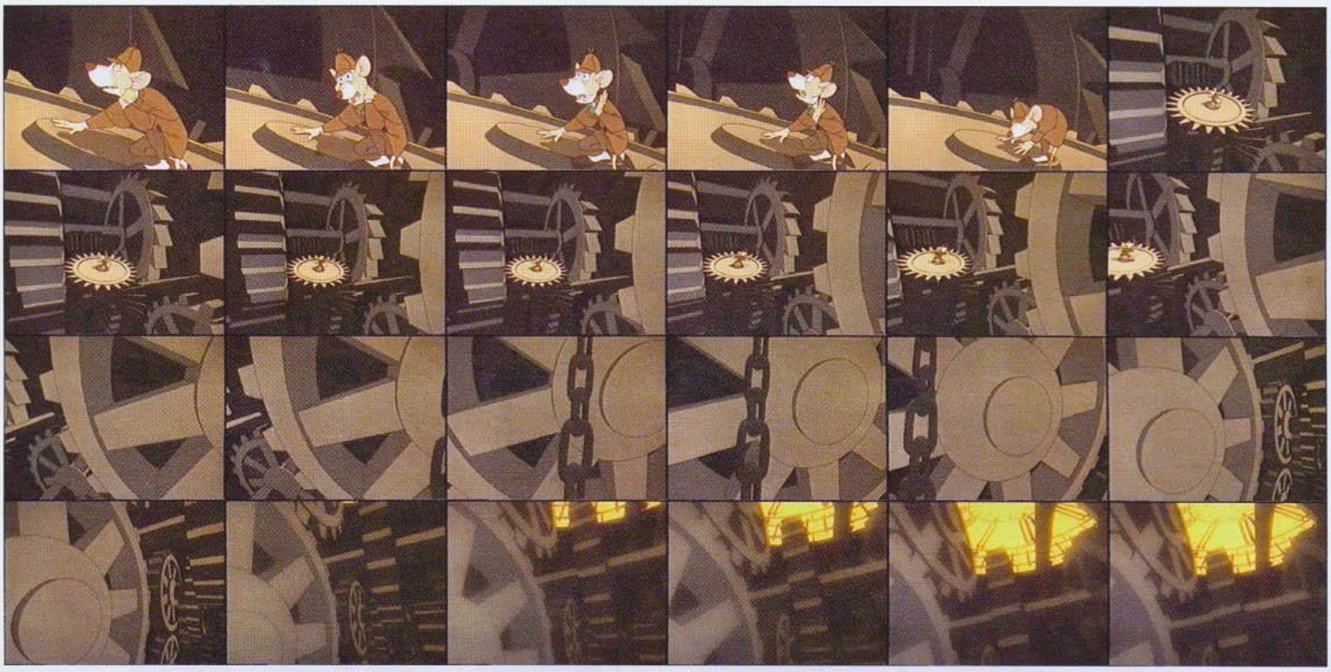
1986 A.D.

- In the film **Flight of the Navigator** the technique of **reflection mapping** is used for the first time, in order to generate realistic reflections of a spaceship.

■ **Figure 2.284**
Playbill and sequence of the film where reflection mapping technique is used on the spaceship.

1986 A.D.

- In the animation film **The Great Mouse Detective, Disney** uses some CG-built objects in a 2D scene, the final sequence inside the Big Ben.



■ **Figure 1.285**
In this scene, which lasts about a minute, the characters are drawn on top of a background scene made entirely in 3D.

1986 A.D.

- **Phillip W. Katz** invents **PKZIP**, a compression program which becomes extremely popular in a short time.

■ **Figure 2.286**
The screenshot of the software's main commands and its creator.

1987 A.D.

- In April **IBM** announces the new series of personal computers, the **PC PS/2**.

With its first personal computer, IBM had created an open machine, in some way putting aside its strong policies on the fact of producing only its self-compatible systems. The enormous success of IBM was due to this approach, which had stimulated many component producers and software developers to create products destined for IBM PCs. At the same time though, an "open machine" is also easy to clone. Considering IBM prices were certainly not unbeatable, it was obvious that their PCs would be eventually copied in thousands of different ways, sometimes simply by putting lower quality components into the clones, but also via cheaper economical production chains. These clones that would appear, would boast the famous "IBM Compatible" sign, even if, to be honest, the compatibility (and even less reliability) was not always guaranteed.

To fix things, IBM then decides to project a machine, more evolved and difficult to copy, which would re-launch sales. The strategic element of this computer was supposed to be its internal communication channel, which went by the name of "microchannel architecture" (MCA). It was a proprietary system, certainly more evolved than the traditional bus, but with the enormous defect of being completely incompatible with cards in circulation. For this reason, an MCA card, with equal service quality, would always cost much more than its ISA correspondent. Lots of PS/2 models are released, from the smaller PS/2-30 (which did not have the MCA bus and was particularly slow) to models 50, 60, 70, 90. All these versions follow parallel to the evolution of various processors, but are to remain confined to a much more restricted use compared to classic IBM PCs. As well as the distinguishing architecture traits, PS/2s have an operating system for multitasking (and then also graphic) developed by Microsoft for IBM: the OS/2 system. The graphic interface (Presentation Manager), database and communications management is certainly to be better in years to come, compared to what the Windows versions were offering. But OS/2 had a few problems: it was too demanding in terms of memory and resource, compared to what PCs offered at the time. It was an IBM system suffering from the same reliability problems that Windows had, as well as being more expensive. One must not then forget that the initial versions of MS Windows were provided free with the purchase of the computer. PS/2 and OS/2, even with the strong support of IBM, were not going to be able to conquer the market. Many were installed in firms where IBM was a fixed presence, being the provider of mainframes and minicomputers, but in any case by the end of the century, both will disappear inevitably from the market, to be replaced by PCI architecture and more recent versions of MS Windows.

Machine type: **PS/2 based on Intel 8086 processors, 16 bit** - Clock speed: **8 MHz** - RAM: **640 Kb** - ROM: **40 or 64 Kb with MS Basic - 80** - Drive: **3.5 inch floppy** - Hard Disk: **20 or 30 MB** - OS: **MS-IBM PC DOS version 3.30**
Price: **$2,950**

■ **Figure 2.287**
Two versions of the famous PS/2.

1987 A.D.

- Adobe produces the first version of **Adobe Illustrator**.

■ **Figure 2.288**
Start-up screen and installation floppy for the first version of Illustrator.

1987 A.D.

- **Apple**, in May, announces its **Macintosh II**. It is the first "modular" version of a Macintosh, defined so because it was contained in a standard case. All the previous versions were all-in-one models with black and white monitors. The Macintosh II model permitted users to use other wide color screens, and if one wanted, also more than one monitor simultaneously, thanks to the possibility of mounting more than one graphics card.

Presentazione:	2 marzo, 1987
Dismissione:	15 gennaio, 1990
Costo:	
CPU:	68020
Frequenza CPU:	16 MHz
FPU:	68881
L1 Cache:	256 KB
L2 Cache:	
L3 Cache:	
Bus:	16 MHz
Slot:	6 NuBus

RAM:	1 MB
Memoria:	Max 20 MB, 8 slot RAM da 120 ns
VRAM:	
Video:	640 x 400 monocromatico LCD matrice passiva
Audio:	Uscita audio a 8 bit
Porte:	2 ADB 2 Seriali Autoparlante microfono SCSI Uscita VGA

Comunicazione:	
Floppy Disk:	2 da 800 KB
Hard Drive:	40 MB SCSI
Consumo:	230 Watt
Peso:	11 kg
Dimensione:	A L P 14 x 47.5 x 36.5 cm
Sistema operativo:	System Software 2.0
ROM:	256 KB

■ **Figure 2.289**
Hardware details and Macintosh II photograph.

To fix things, IBM then decides to project a machine, more evolved and difficult to copy, which would re-launch sales. The strategic element of this computer was supposed to be its internal communication channel, which went by the name of "microchannel architecture" (MCA). It was a proprietary system, certainly more evolved than the traditional bus, but with the enormous defect of being completely incompatible with cards in circulation. For this reason, an MCA card, with equal service quality, would always cost much more than its ISA correspondent. Lots of PS/2 models are released, from the smaller PS/2-30 (which did not have the MCA bus and was particularly slow) to models 50, 60, 70, 90. All these versions follow parallel to the evolution of various processors, but are to remain confined to a much more restricted use compared to classic IBM PCs. As well as the distinguishing architecture traits, PS/2s have an operating system for multitasking (and then also graphic) developed by Microsoft for IBM: the OS/2 system. The graphic interface (Presentation Manager), database and communications management is certainly to be better in years to come, compared to what the Windows versions were offering. But OS/2 had a few problems: it was too demanding in terms of memory and resource, compared to what PCs offered at the time. It was an IBM system suffering from the same reliability problems that Windows had, as well as being more expensive. One must not then forget that the initial versions of MS Windows were provided free with the purchase of the computer. PS/2 and OS/2, even with the strong support of IBM, were not going to be able to conquer the market. Many were installed in firms where IBM was a fixed presence, being the provider of mainframes and minicomputers, but in any case by the end of the century, both will disappear inevitably from the market, to be replaced by PCI architecture and more recent versions of MS Windows.

Machine type: **PS/2 based on Intel 8086 processors, 16 bit** - Clock speed: **8 MHz** - RAM: **640 Kb** - ROM: **40 or 64 Kb with MS Basic - 80** - Drive: **3.5 inch floppy** - Hard Disk: **20 or 30 MB** - OS: **MS-IBM PC DOS version 3.30**
Price: **$2,950**

■ **Figure 2.287**
Two versions of the famous PS/2.

1987 A.D.

- Adobe produces the first version of **Adobe Illustrator**.

■ **Figure 2.288**
Start-up screen and installation floppy for the first version of Illustrator.

1987 A.D.

- **Apple**, in May, announces its **Macintosh II**. It is the first "modular" version of a Macintosh, defined so because it was contained in a standard case. All the previous versions were all-in-one models with black and white monitors. The Macintosh II model permitted users to use other wide color screens, and if one wanted, also more than one monitor simultaneously, thanks to the possibility of mounting more than one graphics card.

Presentazione:	2 marzo, 1987	**RAM:**	1 MB	**Comunicazione:**	
Dismissione:	15 gennaio, 1990	**Memoria:**	Max 20 MB, 8 slot RAM da 120 ns	**Floppy Disk:**	2 da 800 KB
Costo:		**VRAM:**		**Hard Drive:**	40 MB SCSI
CPU:	68020	**Video:**	640 x 400 monocromatico LCD matrice passiva	**Consumo:**	230 Watt
Frequenza CPU:	16 MHz			**Peso:**	11 kg
FPU:	68881	**Audio:**	Uscita audio a 8 bit	**Dimensione:**	A L P 14 x 47.5 x 36.5 cm
L1 Cache:	256 KB	**Porte:**	2 ADB 2 Seriali Autoparlante microfono SCSI Uscita VGA	**Sistema operativo:**	System Software 2.0
L2 Cache:				**ROM:**	256 KB
L3 Cache:					
Bus:	16 MHz				
Slot:	6 NuBus				

■ **Figure 2.289**
Hardware details and Macintosh II photograph.

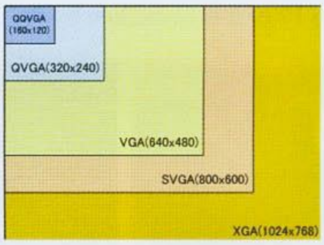
1987 A.D.

- **Microsoft** releases the second version of **Windows, 2.03**.
Features: windows are finally overlappable and resizable, new controls. Not many programs and rather basic structure, but it is included for free with lots of PCs, as well as MS DOS which remains the main operating system.

■ **Figure 2.290**
Opening logo of the program and graphic interface of Windows v2.0.

1987 A.D.

- **IBM** introduces the graphic standard **VGA** for its PS/2 computers. VGA stands **V**ideo **G**raphics **A**rray. It allows graphic resolutions of up to 640 horizontal pixels and 480 vertical pixels and a maximum of 256 simultaneous colors, chosen amongst 262,144 available ones. Unlike the previous standards, which use digital monitors, VGA requires analogical ones, consenting the visualization of continuous blending.
Initially VGA chips were built on PS/2 motherboards but following their success and growing demand, IBM thought well to create dedicated graphic cards, where the integrated VGA circuits would be contained.



■ **Figure 2.291**
The VGA standard is still used nowadays and is the minimum required for a modern PC.

1988 A.D.

- Parametric elaboration of three-dimensional objects takes a huge step forward thanks to the pioneering software **Pro/Engineer** by the firm Parametric Technology Corp. It is a radically new approach amongst design software which maintains the relation of components and features when a model is modified.

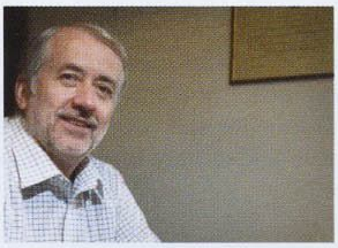
1988 A.D.

- In the film **Willow**, morphing effects are used for the first time. The relative scene is the one where Willow tries to give Fin Raziel back his human appearance: a complex six-stage metamorphosis, where Patricia Hayes goes from being a goat to being an ostrich, then a peacock, a turtle and finally a human being.

■ **Figure 1.292**
Sequence taken from the film "Willow".

1988 A.D.

- **Leonardo Chiariglione**, from the CSELT in Turin, Italy, founds the MPEG (**M**otion **P**icture **E**xperts **G**roup), which will give birth to a vast number of audio and video compression standards, such as **MPEG-1 Audio Layer III**, better know as MP3.



■ **Figure 2.293**
The father of MP3, Leonardo Chiariglione.

1988 A.D.

- Internet has its first crisis with the Internet Worm, a self-replicating program which spreads all through the Web via e-mails, at the time still known as Arpanet. The worm goes down in history with the name of the **Morris Worm**, from the name of its creator, the student **Robert Morris Jr.** of the Cornell University. As a consequence of this crisis, the CERT is founded (**C**omputer **E**mergency **R**esponse **T**eam).

■ **Figure 2.294**
The Morris Worm is based on one of the most basic attack forms, buffer overflow, and strikes about 6,000 hosts out of 60,000 on the net.

1988 A.D.

- **MS-DOS 4.0** is released.

1988 A.D.

- The **Cray Y-MP** is released, the successor of the previous model Cray X-MP. The Y-MP maintains its compatibility with its predecessor, but extends its features, passing from 24 to 32 bits. It is made up of 2, 4 or 8 vector processors, each one with a 167 MHz clock. Peak performance is around 330 Megaflops per processor. It can possess a memory of 128, 256 or 512 SDRAM. In 1990 it is overtaken by the Y-MP Model E and later by the Y-MP M90.

■ **Figure 2.295**
In 1992 Cray launches the new model Y-MP EL.

1988 A.D.

- In June IBM announces the new series of computers, easy to use and designed for small and medium-sized companies: the IBM **Application System/400**, also called the **AS/400**. Contemporarily to the announcement, both IBM and its business partners present over 1,000 software bundles, carrying out the biggest simultaneous promotion in the history of computer trade. The first AS/400 family, which is to be followed by many others, includes 6 processor models, a increase capacity of 24 times the initial amount of memory, 48 times the disk memory, and 10 times the overall performance, measured in terms of the elaboration of commercial transactions by the hour.
The performance compared to System/38 is twice as much, and 5 times as much as System/36.
At the time the first system is sold, 2,500 applications are already available. In the meantime IBM had already sold more than 250,000 S/34, S/36 and S/38 systems all over the world, establishing its leadership as the most widely installed computer series in industries.

■ **Figure 2.296**
The complete AS/400 family.

1988 A.D.

- The concept of portable computers had never been properly defined in the eighties: each producer would launch its own particular model, which would be completely different to others in shape and concept. The **Amstrad PPC 512** is an unmistakeable computer: it is effectively portable, small and compact, with a large plastic handle on the outside. The PPC 512 has a keyboard with a standard configuration which is identical to that of any other computer and determines the rectangular shape of this model: a comfortable solution but also difficult, never to be re-proposed in the future. Inside it has a pull-out LCD monitor and two disk drives. The PPC 512 is worth being remembered for an important reason: it was the first portable sold in Italy for little more than a few million of the Italian lira, although considering its specs, it deserved a lot more.

The practical nature of the Amstrad PPC 512 was without equals: as well as power supplied via transformer or batteries, it could be easily connected to a car's cigarette lighter, allowing one to carry on working also when not at the office or at home.

Processor: **Intel 8088** - RAM: **512 Kb** - Audio: **Beeper** - OS: **MS-DOS 3.X** - Disk: **2 x 3 1/2"** - Weight: **10 Kg**

■ **Figure 2.297**
The Amstrad PPC 512.

1988 A.D.

- The new series of **Motorola 32 bit RISC 88000** offers a speed equivalent to **17 million instructions per second**.

The project was born toward the beginning of the Eighties. Originally called **78000** in honour of the famous 68000 processor series, the project faced many problems and delays and when it was finally presented in April 1988 it had also changed name. Like the previous 68000, the 88000 was considered a very "clean" project. It was completely 32-bit. There was a complete separation between data and address both in the bus and the cache. It was endowed with a small but powerful set of instructions.

The first implementation of the 88000 family was the CPU **88100**, which had an integrated FPU. Following the same path, the **88200** was released, which included a MMU and a cache manager. Creating with the first series of 88000 is extremely expensive, as one must also use many chips in order to so. In fact the CPU, did integrate electronic parts to simplify the construction of the motherboard. This is one of the reasons of the scarce success of the 88000 series. Motorola also produced a series of motherboards in order to permit a faster construction of systems based on the 88000. These motherboards were known as the **MVME** series and later as the **900** series projected with the intention of being included in Racks for industrial use. This series came with a connection to permit communication between the buses of all the units.

At the end of the eighties lots of firms were awaiting developments of the 88000 series. Companies like Apple Computer or NeXT expected to be able to use the 88000 processors as an evolution of the 68000. But in the end they gave up this project. The only case of the use of the 88000 in a project of a certain importance is the Data General AViiON series, although it was abandoned in 1995, when the DG firm decided to use Intel processors. The other well-known machine that used 88000s was the Japanese quad-processor **OMRON luna88k** which was used for a brief period by the Carnegie Mellon University in order to develop the Mach kernel project. There were other projects, but none of them had a spreading influence. Motorola tried to make the project gain popularity with the 88open work group, similar to what Sun Microsystems had don with the SPARC project. But the initiative was not a success. At the beginning of 1990, Motorola teamed up with the AIM alliance, born to create a new RISC architecture based on the IBM POWER architecture.

■ **Figure 2.298**
Some characteristics of the 88000 family were included in PowerPC processors in order to allow users to update with almost no trouble at all. At that point, as soon as possible Motorola stopped the production of the 88000 series.

1989 A.D.

- In the film **The Abyss**, water is simulated realistically for the first time. Using the Alias/2 animation software on the 4D/120 workstation by SGI, Pixar's Renderman on the Pixar Image Computer, and data obtained from Cyberware scanning, the **ILM** creates the water tentacle for The Abyss. The storyboards are scanned and imported into PixelPaint by SuperMac and modified using a software product which is still being developed at that time: Photoshop.

■ **Figure 2.299**
The playbill, and the famous scene of the encounter between the main character and the aliens, beings made of water.

1989 A.D.

- In the film **Indiana Jones and the Last Crusade**, the ILM fulfils its first completely digital compositing.

■ **Figure 2.300**
Images from the film Indiana Jones and the Last Crusade.

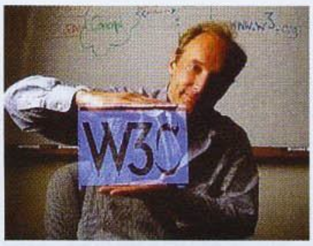
1989 A.D.

- In Genevre, CERN employee, **Tim Berners-Lee**, invents **HTM** and baptises his interconnection project with the name **W**orld **W**ide **W**eb.

He also wrote the first version of a document formatting language with the ability to have hypertextual links, known as HTML. The specific initials for URL, HTTP and HTML were later perfected and discussed by a vast community of programmers and users.

In 1993 Tim Berners-Lee is interviewed by the TG1 news, of the RAI. During the interview, his direct superiors are questioned about the chances of the CERN promoting the WWW idea and its industrial promotion, also by means of special research funds from the European Commission. The CERN director, an Italian physicist, stated that he did not believe it to be the prerogative of CERN, to promote the idea, however brilliant. Tim Berners-Lee, in his book, remembers how difficult it was for him to refuse the request of MIT to host his new creation at the renowned LCS. It is true that CERN did not offer Tim Berners-Lee much choice of refusing the offer of Mike Dertouzos for MIT.

■ **Figure 2.301**
The economical history of the last 20 years could perhaps have evolved differently if only the CERN authorities had shown a different awareness regarding the future.



1989 A.D.

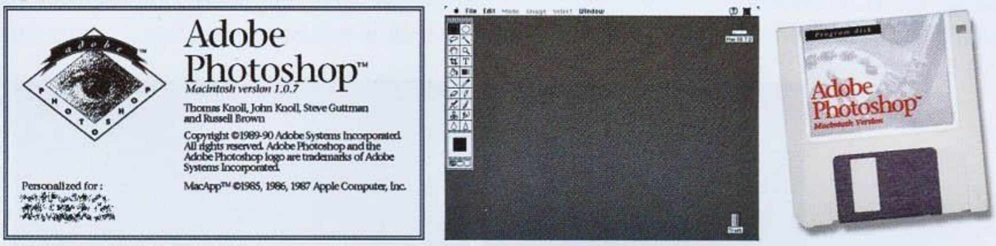
- Motorola presents the personal cellular phone **Micro-TAC**.

■ **Figure 2.302**
Images of the Micro-TAC.

1989 A.D.

- The first **Photoshop** version is produced by the Knoll brothers, sons of a photographer and authors of the program, with the aim of making their father's work easier. It is to be purchased by Adobe.

■ **Figure 2.303**
Opening screen of the first Photoshop version, starting screen and installation floppy.



1989 A.D.

- The first version of **Mental Ray** is produced. It is a rendering engine based on raytracing and its instruction set can be compared to PRman, the rendering compiler by Pixar.

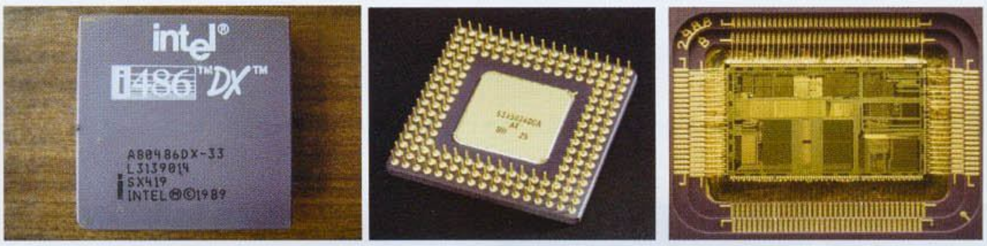
1989 A.D.

- Intel releases the 32 bit **80486** processor. It contains the equivalent of 1.25 million transistors and is able to carry out 20 million instructions per second. From the point of view of the software, the i486 was very similar to the i386, apart from the addition of a few instructions.

From the point of view of the hardware instead, this processor is a very innovative: it possesses a unified memory cache for data and instructions, yet another floating-point calculation unit (optional FPU, only included in the DX version) and an improved bus interface unit. Moreover, in optimal conditions, this processor can carry out an instruction every clock cycle. These improvements allowed the i486 to offer a level of performance almost twice as high as its predecessor, at the same clock frequency. In any case, some of the lower-priced models, especially the SX versions, were effectively inferior to some of the fastest i386 models. In April 1989, a 25 MHz version was introduced; a 33 MHz one in May 1990 and a 50 MHz one in June 1991.

Much like the Intel 80386, its predecessor, the Intel 80486 was released in many versions, amongst which:

- **Intel 80486SX** - an i486DX with one FPU disabled, although the first versions were simply normal i486 models with malfunctioning FPUs. Later the FPUs were removed from the die so as to reduce the surface and consequently also the cost.
- **Intel 80486DX** - as above, but with a working FPU.
- **Intel 80486SX2** - the internal clock of the processor is twice the clock of the external bus, FPU is disabled.
- **Intel 80486DX2** - the i486SX2, but with FPU enabled.
- **Intel 80486SL** - an i486DX with a circuit for energy management, used in portable computers.
- **Intel 80486SL-NM** - an i486SX with a circuit for energy management.
- **Intel 80487** - an i486DX with a slightly different attachment to be used on systems based on i486SX as an FPU.
- **Intel 80486 OverDrive** - an i486SX, i486SX2, i486DX2 or i486DX4. Described as processors for upgrades, some models had different attachments or voltage management abilities compared to ordinary chips of the same speed.
- **Intel 80486DX4** - made to perform at triple clock speed (not quadruple as many often believe). The range of clock speeds included 16, 20, 25, 33, 40, 50, 66, 75 and 100 MHz, although the 100 MHz versions could be unstable.



■ **Figure 2.304**
Some photographs of the Intel 80486 processor.

1989 A.D.

- **Pixar** begins selling their famous rendering product **Renderman**.

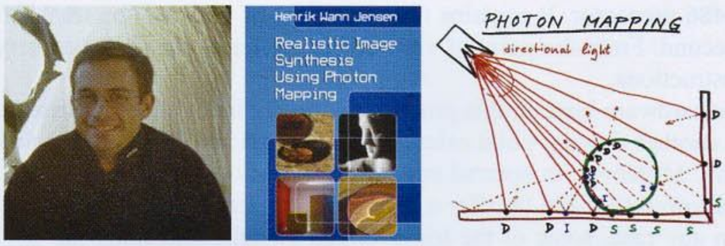
■ **Figure 2.305**
Renderman logo.

1989 A.D.

- A researcher named **Henrik Van Jensen** invents **Photon Mapping**. Officially published in 1996 with his publication by the title of "Global Illumination Using Photon Maps", the algorithm calculates the GI in two steps. The first consists of emitting photons from a light source. These bounce around the scene until they consume their energy. Every time they are deflected though, as well as losing intensity, they gain color, which is carried each time, creating a Photon Map.

The second step, instead, samples each pixel of the render, like a normal ray-tracer system, and then calculates the contribution of each photon for the coloring of the pixels.

As well as eliminating problems caused by coupling radiosity and raytracing, Photon Mapping has some pleasant surprises. Thanks to its nature and to the possibility of passing through matter with its particles, photons are extremely useful to simulate new effects, for instance caustic or translucent ones, such as candles, marble or human skin. Another important factor not to be underestimated is that Photon Mapping is a system which is not based on geometry. If a room is formed by six polygons only (the four walls plus the ceiling and floor) or six hundred of them, the quality does not change.

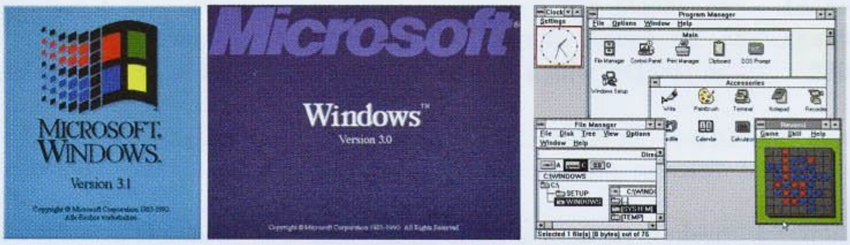


■ **Figure 2.306**
- Henrik Wann Jensen.
- The cover of his book.
- Sketches by the author.

1990 A.D.

- **Microsoft** distributes its new operating system, **Windows 3.0.** In actual fact, it is not correct to define it an operating system, as Windows 3.X was actually a graphic "operating environment" to be installed on DOS.

Windows 3.0 was the first version of Windows to obtain a good level of commercial success, allowing Microsoft to compete with the Apple Macintosh and Commodore Amiga computers in the league of graphic interfaces. This version included a reorganized user interface, as well as technical improvements to permit the use of the functionality of Intel 80286 and Intel 80386 processors. Programs that did not make use of the graphic interface could be executed in a window, making the system usable as a rough multitasking platform for DOS applications. The MS-DOS executive shell was replaced with Program Manager and File Manager and the Control Panel was introduced, which included a limited management of the color scheme of the interface. A limited quantity of applications was included, such as a simple text editor, (Notepad), a word processor (Write), a macro manager and a calculator.



■ **Figure 2.307**
Windows 3.1, released on 18th March 1992, added basic support for multimedia, audio input/output and an applet for the CD player, as well as TrueType fonts for desktop publishing.

1990 A.D.

- The first version of **3D studio** is released. It was already in the phase of being studied in 1988; it was called THUD because of the fact the main programmer was called Tom Hudson. In 1989, at the SIGGRAPH in Boston, Autodesk reveals a new PC based animation software called Autodesk Animator.

1990 A.D.

- In the film **Total Recall**, a new motion capture system is used for the creation, in CGI, of the animated sequence of two characters seen through X-rays.

■ **Figure 2.308**
The playbill and an image from the film "Total recall".

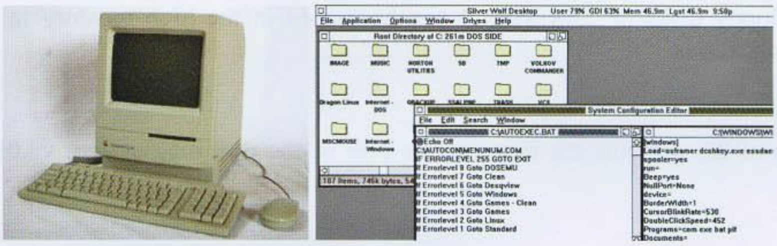
1990 A.D.

- In the film **Die Hard 2: With a Vengeance**, a matte painting is digitally modified for the first time.

■ **Figure 2.309**
Image portraying the backgrounds used in the film.

1990 A.D.

- Apple Computer Inc. releases the **Macintosh Classic**. The demand for another "All-in-one" Macintosh, after the popular Macintosh Plus and Macintosh SE, moved Apple to create the Macintosh Classic. The limitations of the shape of the case, hindered important innovations in this model. An interesting attribute was the opportunity to run the system from a ROM disk using a "Command - Option - X" key combination during start up. Probably the best quality of this model was its cost. The basic model cost 999 dollars and the 49 MByte Hard Disk version cost 1,499 dollars.



■ **Figure 2.310**
The Macintosh Classic was the first Apple product to cost less than 1,000 dollars.

Presentazione:	15 ottobre, 1990
Dismissione:	14 settembre , 1992
Costo:	$ 999
CPU:	68000
Frequenza CPU:	8 MHz
FPU:	
L1 Cache:	
L2 Cache:	
L3 Cache:	

Bus:	8 MHz
Slot:	1 SE PDS
RAM:	1 MB
Memoria:	Max 4 MB, 2 slot RAM da 120 ns
VRAM:	
Video:	9" monocromatico 512x384
Audio:	Uscita audio a 8 bit
Porte:	ADB 2 Seriali Presa cuffie

	Autoparlante SCSI
Comunicazione:	
Floppy Disk:	1.44 MB
Hard Drive:	40 MB SCSI
Consumo:	76 Watt
Peso:	7.3 kg
Dimensione:	A L P 33.5 x 24,6 x 245, cm
Sistema operativo:	System Software 6.0.7

1990 A.D.

- In Italy the **ETACS network is activated** at 900 MHz. In a short while the SIP becomes the top European cellular operator as far as the number of customers is concerned.

1991 A.D.

- In this year, Apple manages to have a reasonable success with the construction of the first version of a PowerBook: the **PowerBook 100**, later followed by the more powerful **140** and **170** models.
In the same year, the first **Macintosh Quadra** models start coming out, mod. 700 and 900, with Motorola 68040 processors at 25 MHz. These are the first Apple Mac computers sold in tower format. The price is 6,000 dollars.

■ **Figure 2.311**
- Photos of the first PowerBook.
- The Quadra Model 700.

1991 A.D.

- **Disney** and **PIXAR** reach a historical agreement. They establish the fulfilment of 3 animation films, including the famous Toy Story.

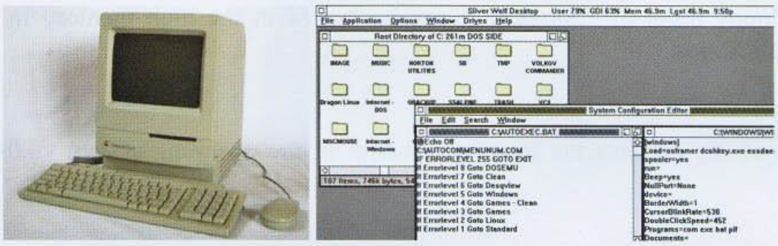
1991 A.D.

- The **ILM** produces **Terminator 2**. For the first time models and photorealistic animations are created for a CGI character. The director relies on these for the T-1000 character, made of liquid metal and able to shape-shift; the character was modelled, animated, rendered and modified with morphing effects by ILM.

■ **Figure 2.312**
Images from the film "Terminator 2".

1990 A.D.

- Apple Computer Inc. releases the **Macintosh Classic**. The demand for another "All-in-one" Macintosh, after the popular Macintosh Plus and Macintosh SE, moved Apple to create the Macintosh Classic. The limitations of the shape of the case, hindered important innovations in this model. An interesting attribute was the opportunity to run the system from a ROM disk using a "Command - Option - X" key combination during start up. Probably the best quality of this model was its cost. The basic model cost 999 dollars and the 49 MByte Hard Disk version cost 1,499 dollars.



■ **Figure 2.310**
The Macintosh Classic was the first Apple product to cost less than 1,000 dollars.

Presentazione:	15 ottobre, 1990
Dismissione:	14 settembre , 1992
Costo:	$ 999
CPU:	68000
Frequenza CPU:	8 MHz
FPU:	
L1 Cache:	
L2 Cache:	
L3 Cache:	

Bus:	8 MHz
Slot:	1 SE PDS
RAM:	1 MB
Memoria:	Max 4 MB, 2 slot RAM da 120 ns
VRAM:	
Video:	9" monocromatico 512x384
Audio:	Uscita audio a 8 bit
Porte:	ADB 2 Seriali Presa cuffie

	Autoparlante SCSI
Comunicazione:	
Floppy Disk:	1.44 MB
Hard Drive:	40 MB SCSI
Consumo:	76 Watt
Peso:	7.3 kg
Dimensione:	A L P 33.5 x 24,6 x 245, cm
Sistema operativo:	System Software 6.0.7

1990 A.D.

- In Italy the **ETACS network is activated** at 900 MHz. In a short while the SIP becomes the top European cellular operator as far as the number of customers is concerned.

1991 A.D.

- In this year, Apple manages to have a reasonable success with the construction of the first version of a PowerBook: the **PowerBook 100**, later followed by the more powerful **140** and **170** models.
In the same year, the first **Macintosh Quadra** models start coming out, mod. 700 and 900, with Motorola 68040 processors at 25 MHz. These are the first Apple Mac computers sold in tower format. The price is 6,000 dollars.

■ **Figure 2.311**
- Photos of the first PowerBook.
- The Quadra Model 700.

1991 A.D.

- **Disney** and **PIXAR** reach a historical agreement. They establish the fulfilment of 3 animation films, including the famous Toy Story.

1991 A.D.

- The **ILM** produces **Terminator 2**. For the first time models and photorealistic animations are created for a CGI character. The director relies on these for the T-1000 character, made of liquid metal and able to shape-shift; the character was modelled, animated, rendered and modified with morphing effects by ILM.

■ **Figure 2.312**
Images from the film "Terminator 2".

1991 A.D.

- A German institute, the **Fraunhofer Institut**, develops a file compression technique which reduces the size of a CD audio track to one tenth of the original size. The technique is well met by the Motion Picture Experts (MPEG) as an official standard and is baptized MPEG 1 Audio Layer 3, or less formally **MP3**.

1991 A.D.

- Microsoft releases **MS-DOS 5.0**, with a graphic shell, a full-screen editor, Unformat and Undelete utilities, a task swapper, support for 3.5" floppy disks of 2.88 MB. It can be loaded in the high memory in order to leave more conventional memory for applications.

1991 A.D.

- Linus Torvalds, at the age of 22, presents the first version ever of the operating system v0.01 which will later be named **Linux**.

1992 A.D.

- The first simulation of human skin is applied in the film **Death becomes her**, as well as improvements in morphing effects in the sequence where the main character twists her neck.

■ Figure 2.313
Images from the film "Death becomes her".

1992 A.D.

- The first version of Open GL is published (SGI). **Open GL** (**O**pen **G**raphics **L**ibrary) is a specification which defines an API for multiple languages and platforms in order to write applications which can produce 2D and 3D graphics. The interface consists of 250 different function calls, which can be used to design complex three-dimensional scenes starting from simple primitive ones. It is used for development in the videogame industry, where it competes with Direct 3D on Microsoft Windows, in CAD applications, virtual reality and CAE.

■ Figure 2.314
Open GL is the effective standard for 3D graphics in UNIX environments.

1992 A.D.

- **IBM** releases **OS/2 2.0**, considered to be a technically revolutionary operating system, to the point that it is crowned with many awards in its field. It allow multitasking with a maximum of 240 separate DOS sessions, as well a programs written for Windows and can manage up to 4 GB of memory. OS/2 had originally been developed by Microsoft and IBM together, as a successor to DOS; Windows was supposed to be a bridge solution between DOS and OS/2 during the preparation of OS/2. Later Microsoft decided to sustain its own product (Windows) and IBM carried on with the development of OS/2 separately.

Bestowed with incredible stability, technically superior to the competition of Microsoft, but less prevalent because of the high demand for memory resources of the first versions and also with a poor amount of software. It never managed to catch up on the initial disadvantage in popularity amongst users, not even when IBM, in a promotional manoeuvre, tried distributing improved versions 3.0 and 4.0 for free. OS/2 still managed to find itself a niche in the bank sector, where it still has a certain base installed.

■ Figure 2.315
Currently the development of OS/2 by IBM is frozen and support for this operating system on the part of the Company has been discontinued on 31st December, 2006.

1992 A.D.

- The **Visual Station** was developed for virtual reality game applications in game arcade centres. Its features included a visor with independent screens for each eye, four loudspeakers and a tracking system for head and body movements. One of the games was called **Dactyl Nightmare**, a virtual reality videogame by Horizon Entertainment, which submerges the player into a surreal world made of chess boards.

■ **Figure 2.316**
The machine and a screenshot from the game Dactyl Nightmare.

1992 A.D.

- **In Italy GSM covering** of the motorway alignments Milan-Naples and Turin-Venice begins officially.

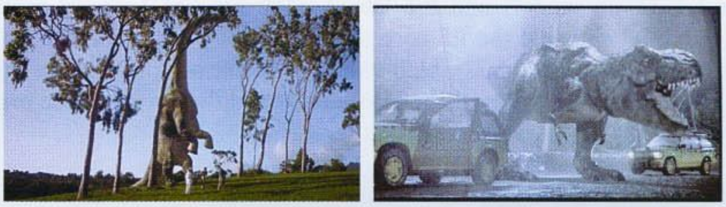
1993 A.D.

- The first real full-length film completely made on a computer is **Veggie Tales**, created by **Big Idea Production**, and not, as is commonly believed, Toy Story by Pixar. The characters are animated vegetables, the reason for this choice being the limited capacity of CG systems at the time. The creation of humanoid Images with legs and articulations would have been difficult and time-consuming for a film this long.

■ **Figure 2.317**
After this first video, in the following years more than 35 million copies. Veggie Tales is a series of books and videos for 12 year-olds.

1993 A.D.

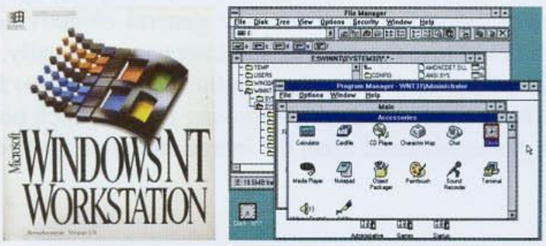
- In **ILM's Jurassic Park**, directed by Steven Spielberg, for the first time photorealistic people and animals are computer generated in a cinematographic film.



■ **Figure 2.318**
Images taken from the film "Jurassic park".

1993 A.D.

- **Microsoft** releases **Windows NT**, destined to be used as a network server OS. Windows NT was originally based on the OS/2 NT project, a team effort between IBM and Microsoft. The collaboration ceased and IBM continued to commercialize the previous version of OS/2 whereas Microsoft renamed its own version as Microsoft Windows NT, including the APIs from Windows 3.X, adapted for 32 bits, as high level APIs. The graphic interface is that of Windows 3.X. It was released on the market on 27th July 1993 and was not a great success, but it can be considered the father of modern Windows systems. The next version, 3.5, soon after followed by 3.51, marked the rise of Microsoft in this area of the market, which in the following years it would come to dominate, at the expense of Novell. The first version is 3.1, unlike the more traditional version 1.0, with the aim of stimulating buyers to associate it with Windows 3.1, which was selling a remarkable amount of copies. Versions 1.0 and 2.0 of Windows NT never existed. Windows NT offers pre-emptive multitasking with the graphic interface of Windows 3.1 and is able to execute applications for Windows, DOS and OS/2 (16 bit), as well as 32 bit programs developed specifically for Windows NT. In 1996, exactly one year after Windows 95, of which it is not, however, the successor, Windows NT4 comes out, dedicated to servers and workstations.



■ **Figure 2.319**
Packaging and screenshot of the software.

1993 A.D.

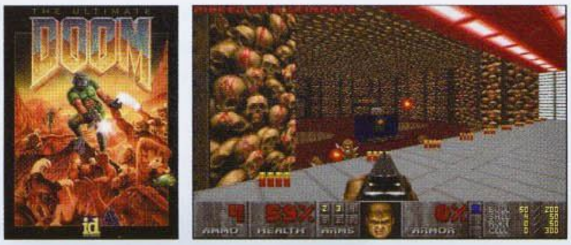
- **Id Software** releases the videogame **Doom**, an evolution of Wolfstein 3D. It becomes exceedingly popular, not only because of the simple plot, the abundant violence and the action taking place with a first-person viewpoint, but also because of its three-dimensional graphics which were very sophisticated compared to games of the same period.

The reason for the great success of Doom is attributed to its highly realistic appearance for 1993 standards. The game distinguishes itself greatly from the rather bland Wolfenstein 3D and the difference is typically associated to that between a black and white film and a color film. Compared to Wofenstein 3D, Doom introduced the following changes in the management of the virtual world to be explored:

- Difference in height between levels (the rooms in Wolfenstein 3D are all on the same floor)
- Non-perpendicular walls
- Texture mapping applied to all surfaces
- Variation in the degree of luminosity (so that some levels are sunken into a sinister shadowy atmosphere)

Unlike its forerunner, Doom levels are extremely interactive: one can use lifts, mobile bridges and observe other types of structures in movement.

▪ **Figure 2.320**
The level of graphic realism is comparable to that of the sound effects. From MIDI it has passed to stereo audio, much more realistic and embracing.



1993 A.D.

- **Intel** presents the 32 bit **Pentium** processor. The Pentium is a microprocessor belonging to the fifth generation of processors with x86 architecture. It made its first appearance on 22nd March 1993. It followed Intel 80486, and was initially supposed to be called Intel 80586, or i586, but then the name was changed in Pentium (the prefix Pent means "five" in Greek), because numbers cannot be registered as a trademark whereas words can. The term i586 is in any case used in programming environments to refer to all the first Pentiums and compatible processors produced by competitors of Intel. The project had been named "P5" in the stage of development.

Pentiums offered almost twice the performance per clock cycle compared to a 486. The fastest i486 models would equal a speed close to the earliest Pentium models and the late Am486 by AMD would reach a Pentium 75. Later the **Pentium MMX** was introduced, which used the P5 core and the production process at 0.35nm. Also, it included the **MMX** instruction set, made up of 57 instructions, which allowed greater performance in treating multimedia data. The software had to be optimized to be able to exploit these instructions and the performance improvement of the P55C was mainly due to the fact of doubling the memory cache so that it reached 32 Kb.

▪ **Figure 2.321**
The first Pentium were released with speeds of 60 and 66 MHz; later they became available in versions at 75, 90, 100, 120, 133, 150, 166, 200 and 233 MHz.

1993 A.D.

- Microsoft releases **MS-DOS 6.0** which, compared to version 5.0, adds the compression program DoubleSpace, an enhanced backup and restores system, an antivirus, and a defragmenter. Many of these utilities are included with a license from Central Point Software or Peter Norton Computing instead of having been written by Microsoft.

1993 A.D.

- **Broderbund Software** publishes **Myst**, elevating videogame standards in terms of plot and imagery. The creators of this adventure employ reflection, refraction and shadows to render, through ray-tracing, the 2,500 images that make up the game. It is a revolutionary game for the times: the general objective of the developers was to create a genuine parallel world, where players could lose themselves and interact realistically.

In obscure circumstances, the player (called Stranger in the Myst universe) finds a strange book called Myst. Browsing through it, at a certain point he is confronted with a moving image which portrays an airborne view of a strange island. Without thinking, the player places his hand upon the image and is catapulted inside!

Inspecting his surroundings, he notices that he is in the island's small harbour; in the background the sound of water and seagulls can be heard. Unable to get back to his own world, the player decides to take a look around…

■ **Figure 2.322**
Myst sold over 10 million copies, and for a long time it was the videogame bestseller of all history.

1994 A.D.

- Two US students, David Filo and Jerry Yang found **Yahoo, Inc**.

It started as a student's hobby and then in February 1994 they created a deposit of links able to keep track of their personal interests on the internet. Soon the list became too crowded and chaotic, so they decided to divide it into categories, which soon suffered the same problem, leading to yet another subdivision into categories and, in the end, to the main concept beneath the newborn Yahoo! Project. The original name for the site was "Jerry and David's Guide to the World Wide Web". But it was soon replaced with Yahoo!, acronym for "Yet Another Hierarchical Officious Oracle", chosen, according to the two founders, because of its meaning in the English language: uncouth, unsophisticated and graceless. Initially it was hosted on the PCs of its two creators, called Akebono and Konishiki like the two legendary sumo wrestlers. It soon became the chosen reference point for students of the Stanford University, and then a point of reference for the internet community. In Autumn 1994, Yahoo! reached one million daily visits, with 100 thousand single visitors. In 1995, having seen their great success, Jerry and David sought out an investor able to support their project. They came across Sequoia Capital, a well-known society thanks to its previous success with Apple Computer, Atari, Oracle and Cisco Systems. In April 1995 they founded Yahoo! with an initial investment of 2 million dollars. More and more they realised the true potential of their endeavour and began too look for an executive team. They employed Tim Koogle as head of the executive office and Jeffrey mallet as head of the operational office. In autumn 1995 they found other investors and in April 1996 they launched an IPO. The firm, at this point, was made up of 49 employees.
From 1996 until today the firm has gradually grown in size and includes ramifications in the field of communications, internet services and distribution media. It has a visibility of 345 million people per month and is at the pinnacle of the technological computing industry.

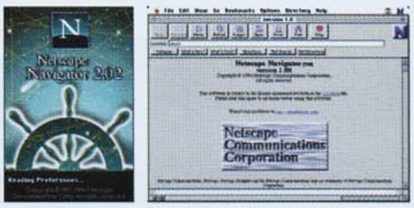
YAHOO!®

■ **Figure 2.323**
Logo of the famous internet site and search engine.

1994 A.D.

- In April, **Jim Clarke** and **Marc Andreseen** found **Netscape Communications**, originally called Mosaic Communications. **The first browser of Netscape** is released and it is an immediate boon for Web surfers.

Once the main web browser, in time Netscape's star started to sink: it has gone from being the most widely used browser in 1988 (thanks to the lack of competition and a few technical innovations such as the SSL protocol) to currently being used by no more than 1% of web surfers. This crisis is due to many factors and mistakes, partly fault of the software development team and of the management of the Netscape Communications Corporation. In 1998 Netscape practically had no competition, Microsoft Internet Explorer has been around since 1993, the year that version 1.0 came out, but at the time there was no chance of it competing with the suite that Netscape Corporation had released, both in terms of functionality and precision in site-rendering. Internet Explorer had reached version 4.0, the same version number that Netscape had reached in the meantime in 1998, but it was not provided with Microsoft Windows and Bill Gates' firm had not yet encouraged its circulation and continued to see it as an application module external to the operating system.
At the beginning of 1997, and then definitively in June 1998 with Windows 98, Microsoft included Internet Explorer within the operating system, to the point that the two products were structurally inseparable, according to the Company. The presence of a readily configured, efficient and user friendly browser in every computer determined the success of Microsoft's browser. Although some users did not yet consider Internet Explorer to be at the level of Netscape, its spreading quickly became unstoppable.



■ **Figure 2.324**
Logo, and screenshot from the interface of the first real browser.

1994 d.c.

- Caldera and SuSE begin the distribution of **Linux**, its kernel having reached version 1.0. **SUSE** is one of the main GNU/Linux distributions, produced in Germany.

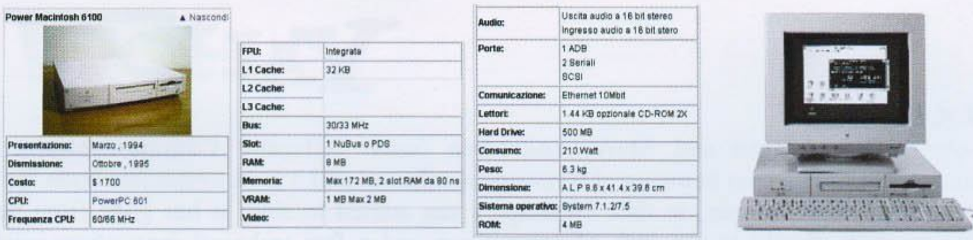
SUSE was founded at the end of 1992 by Hubert Mandel, Roland Dyroff, Burchard Steinbild and Thomas Fehr as a consultation group for UNIX, which, amongst other things, would regularly release software bundles including SLS and Slackware and would also publish Unix and Linux manuals. The first CD version of SLS/Slackware was released in 1994, under the name S.u.S.E. Linux 1.0. Later also the distribution of Jurix by Florian La Roche was included, also based on Slackware, and the first real S.u.S.E. Linux 4.2 distribution in 1996. The name S.u.S.E., later abbreviated as SuSE, was originally the acronym for Software-und-System-Entwicklung (German for Software and System development).

■ **Figure 2.325**
During 1994 distribution changed the name slightly, to "SUSE Linux", all in capital letters.

1994 d.c.

- In March **Power Macintosh 6100** comes out, the first computer with a processor of the **PowerPC** series. It is in fact in 1994 that the new line of personal computers with **PowerPC** processors is born. This processor was developed by Motorola and IBM, and able to guarantee superior performance compared to personals based on Intel processors. The last model of the first series of Power Macintosh was presented in February 1997 and was the Power Macintosh 9600 model.

The Power Mac line of products is dedicated to advanced users, who are in need of high performance and thus must be willing to pay the price. The main users are professionals and university research centres. In the Power Mac series, Apple develops and tries out all its innovations (bi-processor systems, liquid cooling, etc.), some of which are to be introduced in series which are dedicated to a wider market of users, whereas others are to remain confined on the Power Mac, because of their high production cost. Many Power Macs are provided with series of two processors in order to obtain the highest possible performance. The cost factor during the development of these machines is very important, but not the main worry, as with machines for the wide community of users.



Power Macintosh 6100 ▲ Nascondi

Presentazione:	Marzo , 1994
Dismissione:	Ottobre , 1995
Costo:	$ 1700
CPU:	PowerPC 601
Frequenza CPU:	60/66 MHz

FPU:	Integrata
L1 Cache:	32 KB
L2 Cache:	
L3 Cache:	
Bus:	30/33 MHz
Slot:	1 NuBus o PDS
RAM:	8 MB
Memoria:	Max 172 MB, 2 slot RAM da 80 ns
VRAM:	1 MB Max 2 MB
Video:	

Audio:	Uscita audio a 16 bit stereo Ingresso audio a 16 bit stero
Porte:	1 ADB 2 Seriali SCSI
Comunicazione:	Ethernet 10Mbit
Lettori:	1.44 KB opzionale CD-ROM 2X
Hard Drive:	500 MB
Consumo:	210 Watt
Peso:	6.3 kg
Dimensione:	A L P 8.6 x 41.4 x 39.6 cm
Sistema operativo:	System 7.1.2/7.5
ROM:	4 MB

■ **Figure 2.326**
- Hardware features of the Macintosh 6100.
- Photograph of the Macintosh 6100.

1994 d.c.

- For the film **The Lion King**, the CGI department of **Disney Feature Animation** animates herds of 3D wildebeest which are combined with painted backgrounds. Disney reveals the existence of its production and animation system on the **CAPS** computer, carefully supervised and developed with the collaboration of Pixar. CAPS is a software bundle for digital ink-and-paint, compositing and rendering. After an honourable career, with Disney's decision in 2005 to abandon traditional 2D animation, the various CAPS systems are dismantled, leaving only a few models for revisiting old works.

1995 d.c.

- In the film **Casper**, for the first time CGI characters interact with real ones. Casper is probably the first 3D protagonist in a live-shoot film and is definitely the first 3D protagonist animated by **ILM**.

■ **Figure 2.327**
The society created 200 scenes with special effects for the film and many of these integrate semi-transparent characters with real-life backgrounds.

1995 d.c.

- **Dreamworks SKG** is born. The Company was founded by Steven Spielberg (already a director and the co-founder of Amblin Entertainment), Jeffrey Katznberg (earlier head of the animation department for The Walt Disney Company) and David Greffen (founder of Greffen Records, an important record company). It was founded in October 1994, thanks to the funding by Paul Allen (co-founder of Microsoft).

■ **Figure 2.328**
Its entry into cinematographic production began in 1997 with the film "The Peacemaker" by Mimi Leder.

1995 d.c.

- The **Playstation** is introduced in western countries. It is a 32 bit videogame console, produced by Sony. It was launched in Japan on the 3rd December 1994, in the USA on the 9th September 1995, whereas it arrived in Europe on the 29th September 1995. As well as allowing games to be played from CDs, the Playstation can also be used to listen to musical CDs.

■ **Figure 2.329**
The console was so popular that it gave the name "The Playstation Generation" to young people in the nineties.

1995 d.c.

- **Pixar** produces its first long film, **Toy Story**. It is played at the cinema for the first time on the 21st November 1995 in the US and on the 22nd March 1996 in the UK. It will make $356,800,000 throughout the world.

■ **Figure 2.330**
It is in 56th position amongst the most sold films.

1995 d.c.

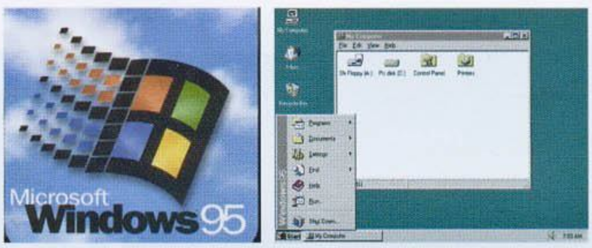
- IBM releases **PC-DOS 7.0**.

1995 d.c.

- In Italy **SIP starts providing GSM commercial service.**

1995 d.c.

- **Microsoft** releases **Windows 95**. In 4 days time over a million copies are sold. Internet Explorer 1.0 is included in the product. The Intel 80386 had already been marketed in 1986, Intel's first 32 bit microprocessor, later followed by the Intel 80486. So in 1995 almost all computers were able to carry out 32 bit codes, although most were still stuck on 16 bits with DOS or Windows 3.1. Microsoft had already releases a 32 bit operating system previously, Windows NT, but it was reserved for professional use and servers, needing a powerful and expensive computer in order to work. Also IBM had its 32 bit system, OS/2, which at the time was in its most successful moment; but also OS/2 required elevated resources, especially in terms of RAM memory which, at that time, was very expensive. Windows 95, instead, could function, although with difficulty, on an 80386 with only 4 Mbytes of RAM, so on computer of a lower level compared to the ones needed to run NT. This was the winning trait that made Windows 95 a commercial success, and also the first real 32 bit operating system "for the masses". It did, however, conserve numerous portions of 16 bit code, derived from Windows 3.1 and MS-DOS. The fact of being a 16/32 bit hybrid was one of the reasons it could function on low-memory systems, but was also the reason for its instability.



■ **Figure 2.331**
The 1995 version of Windows 95 was followed by an updated version, called Windows 95 OSR2, sold only as a pre-installed system on a new computer. It introduced FAT32 and a filesystem capable of handling partitions larger than 2 Gbytes.

1996 d.c.

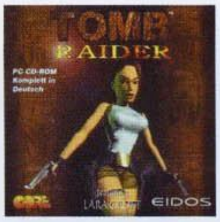
- iD Software, belonging to John Carmack, releases **Quake**, a glorious Game which will remain in history as a milestone of the FPS genre. The Graphic Engine is actual 3D, versatile and powerful, and was the first to offer support for accelerator cards supporting the fist OpenGL drivers, such as the famous 3Dfx. It had quite realistic graphics and real-time illumination. In time various changes were applied: the first version of Quake ran on DOS and was one of the most advanced games for that operating system and offered graphic options regarding high resolution, to be obtained with VESA drivers. It could reach 1280x1024 pixels. The fist Windows version, called WinQuake or Quake version 1.9 came out in March 1997, and offered even better graphic power-ups. The final version was called GLQuake, an improved version of WinQuake and was released towards the end of 1997, offering full support for OpenGL drivers.

■ **Figure 2.332**
The source code for Quake is now distributed under Open Source license.

1996 d.c.

- The videogame **Tomb Raider** is released. The main character, Lara Croft, became extremely popular: it is the first time that a videogame character is considered sexy. The first game was initially released for the Playstation, the Sega Saturn and for PC. The main character was modelled and animated with **3D studio 4**, and the textures were carried out with **Photoshop**.

■ **Figure 2.333**
Lara Croft is an archaeologist with two main traits: she always goes round with shorts and a skimpy vest and has breast which are big enough to defy the laws of nature.



1997 d.c.

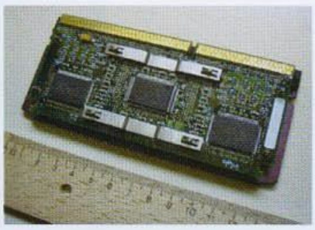
- **The IVA tax on musical CDs** in Italy increases to 20%. In the years before the values were the following:

1985: **9%** - 1991: **12%** - 1994: **13%** - 1995: **16%** - 1997: **20%**

1997 d.c.

- The **Pentium II** is a x86 microprocessor produced by **Intel**. It was based on "P6" architecture developed for the Pentium Pro, initially designed to substitute the first Pentium, but later confined to the market of high-price products. Compared to the Pro, the Pentium II offers better performance in 16 bit code and supports MMX instructions introduced with the later versions of the first Pentium. The first version of Pentium II was called Klamath, clocked at 233 and 266 MHz, produced with a 0.35 μm process. For that period, the amount of heat it produced was enormous. It also had an FSB at 66MHz, an inadequate speed which did not allow the processor to show its full potential. Later in 1997 a 300 MHz version was released. Pentium II models with Deschutes architecture appeared in January 1998 with a speed of 333 MHz. These were produced with a more appropriate process at 0.25 μm, and would heat up significantly less, they were provided with a 100 MHz FSB, which allowed considerable improvements in terms of performance. In 1998 other versions were released with speeds of 266, 300, 350, 400 and 450 MHz. In systems based on Pentium II processors, a new type of RAM was introduced, SDRAM (which replace EDO RAM), and the graphic bus AGP. Unlike its forerunners, the Pentium II did not have an attachment socket, but was enclosed in a covering which had to be inserted into a slot. This compromise allowed Intel to separate L2 caches from the processor, and avoid production problems which had been had with the Pentium Pro, in any case keeping it closely connected to the processor with an extremely fast bus. But this cache, unlike the previous Pentium, worked at a clock speed which was half as fast as the processor speed.

■ **Figure 2.334**
Cheap version of the Pentium II, basically a Pentium II with little or no L2 cache, was called Celeron. The server version was called Pentium II Xeon. In the photo a Pentium II Klamath at 266 MHz.



1997 d.c.

- The first **DVD-ROMS** are produced.

The DVD was the product of the collaboration between some of the major firms in the field of research and consumer electronics: the so-called DVD forum. The institution which was in charge of directing the specifications for the new support was in fact made up of the following firms: Philips, Sony, Matsushita, Hitachi, Warner, Toshiba, JVC, Thomson and Pioneer. The intention was to create a storage format compatible with the large file sizes of digital videos which could be accepted without restrictions by all major producers, thus avoiding all the problems of uncertainty on the market due to competition between formats which had started existing at the time of the introduction of videocassettes for domestic use. The DVD forum establishes three main fields for DVD application:

- DVD-Video, destined to contain films, in substitution of videocassettes.
- DVD-Audio, destined to substituted Audio CDs thanks to a greater fidelity.
- DVD-ROM, destined to replace CD-ROMs.

Both in DVD-Videos and DVD-Audios protection systems able to deter users from copying content. Because of problems in the development of apt protection codes, the DVD-Audio standard seems to be the less lucky of the applications of the DVD. On the contrary, the DVD-Video and DVD-ROM formats appeared on the market from 1997 onwards, obtaining an enormous commercial success.

Later, the DVD forum introduced recordable formats of DVDs. Formalized in 1999, the DVD-R format is the official standard for recordable DVDs. It is divided into "DVD-R for authoring" and "DVD-R for general use" formats. The former are used for the creation of copies of videos protected b copyright, requiring a special DVD writer. They are capable of implementing copy-protection systems. The "DVD-R for general use" format is able to contain any type of material, but is not compatible with protection systems used in DVD-Videos.

In the year 2000, the DVD-RW standard was formalized, with follows in the tracks of the "DVD-R for general use" but with the capacity to be used up to 1000 times (in theory). In the years during which the DVD-R standard was formalized, 2 other formats were marketed for recording on DVD format: the DVD+R (and DVD+RW) by the duo Sony-Philips, and the DVD-RAM format, supported by Matsushita and JVC. These formats differ considerably from the DVD-R in technical terms, although modern readers and writers are compatible with recordable DVDs in any format (with only the DVD-RAM giving a few problems).

Even more recent was the introduction of standards for double-layer DVDs, similar to industrial DVD-9 formats, and with a capacity of 9 GB of information. Also in this case the Sony-Philips team got a head start, releasing the DVD+R Dual Layer (DVD+R DL) in 2002, whereas only later, in 2004, was the DVD-R DL format put into commerce.

Although the industry sector has decreed that DVDs are technologically dead, there is the impression that their use will continue for a period which does not appear to be short. In the meantime there is a terrible battle to decide what will take its place.

The standards suggested to succeed the DVD are Blu-Ray and high definition DVDs (HD-DVD), which are mutually incompatible. One must hope that industries reach an agreement quickly, in order to avoid commercial wars which would make consumers lose their bearings, similarly to what happened at the times of the war between VHS and Betamax formats. In actual fact, the standard which promises a true improvement is almost ready: the HVD, based on the technology of holographic memory, and will be able to contain a little less than 4 Terabytes on every disc.

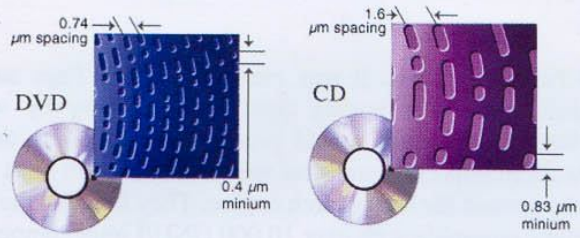


Figure 2.335
Logo of the DVD standard and graphic representation of the differences between DVDs and CDs.

1997 d.c.

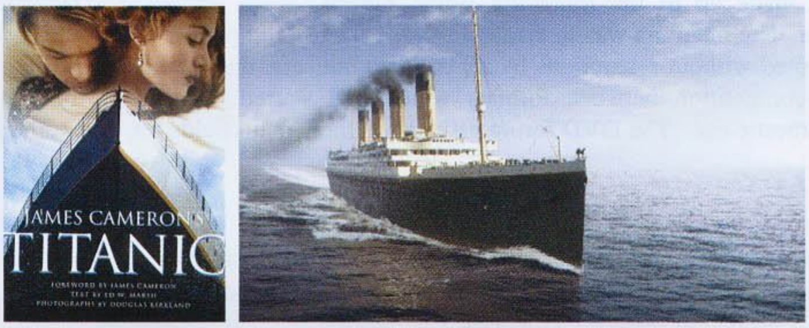
- Many of the 218 scenes with special effects created by **Tippett Studios** for the film **Starship Troopers** show swarms of immense and evil alien creatures.

Figure 2.336
The team made use of Softimage\3D, Dynamation, Amazon 3D paint and Photoshop. Furthermore, a device for data input called "Bug Input Device" allows one to animate the insect warriors in real-time.

1998 d.c.

- **Titanic** is the first drama film to make enormous use of CGI. **Digital Domain**, the most important studio amongst the ones which worked on Titanic, created something completely new by inserting thousands of digitally-created people, animated with the motion capture technique, aboard the ship.

■ Figure 2.337
The playbill of the film and the navigation scene of the Titanic, obviously in CG.



1998 d.c.

- Shawn Fanning, student of Boston College, creates **Napster**, provoking a striking mass-violation of copyrights. The service was named "Napster" from the nickname adopted by Fanning.

Napster was the first large scale peer-to-peer system and became available in autumn 1999. But it was not a pure peer-to-peer, as it used a system of central servers which kept the lists of the connected systems and shared files, whereas the actual transactions would take place directly between users. In fact this is a system which is very similar to instant messaging. In any case, there were already some relatively popular means of sharing files, for example IRC, Hotline and USENET.

Napster was the first to specialise itself only in MP3 files. The result was a system that differentiated itself from others for the possibility it gave of choosing between an amazing amount of music files. Almost immediately, in December 1999, the major record companies started a lawsuit against Napster. This gave Napster an enormous amount of media hype and the user count took off and soared at a peak of 13.6 million users in February 2001.

In July 2001, it was ruled that Napster should shut down its servers because of the continued copyright violations. In September 2002, a bankruptcy court ruled that Napster must liquidate its assets, according to chapter 7 which regulates bankruptcy cases in the USA. Most of the employees were fired and the website was closed, leaving just a banner, "Napster was here".

Roxio is the name of the firm that purchased Napster's assets during the auction, with the aim of using them for a new service on payment. The service was nicknamed "**Napster 2.0**" and released as a Beta Version in New York on 9th October 2003, beginning to fully function on the 29th. Napster 2.0 is not a peer-to-peer service and has little in common with the original Napster, apart from the name and the logo. It was one of the many online legal music services which sprung to life following the tracks of the success had by Apple Computer's "iTunes Music Store".

■ Figure 2.338
All that remains of the real Napster.

napster
WAS HERE

1998 d.c.

- **Google** is born, at Menlo Park, California. It was created by Larry Page and Sergey Brin, who at the time were students at the Stanford University, after having developed a theory by which a search engine based on the mathematical relationships between websites would produce better results than the empirical results used so far. Convinced of the fact that the more important websites were the ones with most links, they decide to deepen the theory within their studies and laid the ground for their search engine. They founded their company in September 1998.

Currently Google uses a machine ensemble with over 10,000 GNU/Linux computers in order to respond to searches and index the web. The cataloguing is done by a program ("googlebot") which regularly acquires new copies of the web pages it knows about. The links of these WebPages are examined to discover new sites and add them to the database. Its index, together with the cache, takes up many Terabytes.

The word Google, derived from **googol**, a term invented by Milton Sirotta (nephew of the famous mathematician Edward Kasner) in 1938, defines the number represented by a one followed by 100 zeros. The use of the word Google reflects the will of society to organize immense quantities of information available on the Web.

■ Figure 2.339
Logo of the incredibly famous internet site.

Google™

1998 d.c.

- The first version of **Maya** is released, well-known 3D software. Maya is the product of three lines of software: Advanced Visualizer by Wavefront in California, Thomson Digital Image Explore (TDI) in France and Alias Power Animator in Canada. In 1993 Wavefront buys TDI and in 1995 Silicon Graphics Incorporated (SGI) buys both Wavefront and Alias. It is quite probably due to pressure put on the market after Microsoft bought Softimage. From 1998 onwards, Alias/Wavefront will stop updating any 3D softwares, including Power Animator, in order to force users to buy Maya.

■ **Figure 2.340**
The cost of the Unlimited version was over 60 million Italian Lira.

1998 d.c.

- The **MPEG-4** is presented. It is the name given to a series of standards for digital audio and video codification, developed by the ISO/IEC Moving Picture Experts Group (MPEG). The MPEG 4 is the standard used mainly for applications such as video-telephone systems and digital television, for the transmission of films via Web and storage on CD-ROMs.

The concept at the base of the MPEG 4 codec (Encoder-Decoder) is quantization. Without going into details, we can quickly define it as the process that permits one to transmit only the variation in the images, through an appropriate compression algorithm.

1998 d.c.

- **Windows 98** is launched, an operating system following Windows 95 and with better support for hardware standards such as USB, MMX and AGP. Other features were FAT 32 filesystem support, multiple monitors, web TV, and the amalgamation of Internet Explorer within the graphic interface (GUI) of Windows, called Active Desktop (ability to setup HTML web pages or interactive/animated objects as a desktop wallpaper).

Windows 98 SE (Second Edition) was released on the 10th of June 1999. It includes other improvements compared to the first version, like Internet Explorer 5, Windows Netmeeting 3, Shared Internet Connection and DVD ROM support. It was criticised for its lack of innovation and because of the fact it was not released as an update for the first version, but, even so, it was a product of success.

■ **Figure 2.341**
Windows 98 was substituted by Windows ME and Windows XP, the first product for home use based on the kernel of Windows NT.

1998 d.c.

- **Apple** introduces **iMac**, which are supposed to go back to being user-friendly computers, as well as having a revolutionary design. It is the first personal computer without disk drives. The distinctiveness of the iMac lies in its mono-block form, much like the first Mac (short for Macintosh), which includes an internal modem, a network card, USB ports, CD-ROM reader, 15 inch color screen and a semi-transparent color case.

Presentazione:	Agosto , 1998	**Bus:**	66 MHz	**Porte:**	2 USB
Dismissione:	Gennaio , 1999	**Slot:**		**Comunicazione:**	Ethernet 10/100
Costo:	$ 1299	**RAM:**	0 MB	**Unita Ottica:**	CD-ROM 24X
CPU:	PowerPC 750	**Memoria:**	Max 256 MB, 2 slot RAM da 10 ns	**Hard Drive:**	4 GB IDE
Frequenza CPU:	233 MHz	**VRAM:**	6 MB SGRAM ATI RAGE	**Consumo:**	80 Watt
FPU:	Integrata	**Video:**	15" RGB 1024 X 768 24 bit	**Peso:**	15.8 kg
L1 Cache:	64 KB			**Dimensione:**	A L P 40.1 x 38.6 x 44.7 cm
L2 Cache:	512 KB	**Audio:**	Uscita audio a 16 bit stereo Ingresso audio a 16 bit stereo	**Sistema operativo:**	System 7.1
L3 Cache:				**ROM:**	1 MB 3 MB caricati in RAM

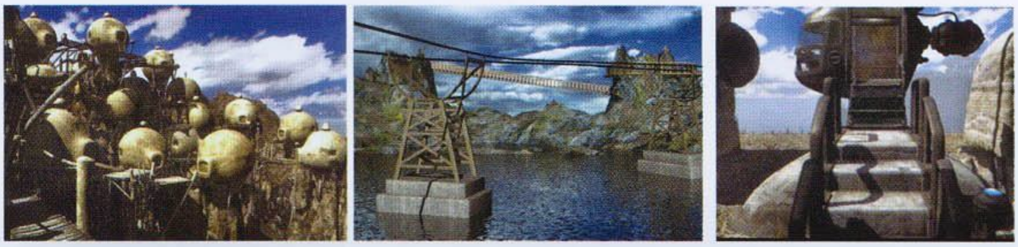
■ **Figure 2.342**
It has been conceived as a personal tool, particularly apt for Internet surfing, with PowerPC G3 processors, advertised as being able to "burn" the Intel Pentium equivalent processors, which iMac will later use itself.

1998 d.c.

- **Cyan** took four years to give birth to **Riven**, the much awaited sequel to Myst. The game's objects have been modelled within **Softimage/3D**, whereas the complex internal illumination was handled by Mental Ray.

Riven starts from the exact point where Myst left off, in the crypt of K'Veer in D'ni where Atrus had been imprisoned until the arrival of The Stranger. The player once again takes on the role of this friend of Atrus. Atrus needs The Stranger's help to free his wife, Catherine who has been trapped by Gehn, Atrus' mad father, who has proclaimed himself God of Riven and reigns tyrannically over its inhabitants…

■ **Figure 2.343**
Some screenshots from Riven.



1999 d.c.

- Webbie Tookay is presented, **the first virtual model**. Made up of elements taken from various real-life models and with a "multiethnic" skin color (more or less coffee-color), she struts elegantly on TV and computer screens in fashion shows and for TV advertisements. It is another step towards realism for digital animation.

■ **Figure 2.344**
Some facial expressions of the virtual model.

1999 d.c.

- **Intel** has reached **Pentium III**.

Katmai

The first version of the Pentium III, called Katmai, was basically a Pentium II built with a process of 250 nm, with the addition of SSE instructions and an improved control of L1 memory cache of 512 Kb. It was initially sold with a speed of 450 MHz and 500 MHz with a 100 MHz BUS. Katmai used the same covering to be inserted into slots as the Pentium II, Slot 1.

Coppermine

The second version, called Coppermine, built at 180 nm, had a L2 cache memory reduced to 256 Kb but integrated at full speed, which improved performance compared to Katmai. Feeling the pressure of competition from AMD with its Athlon Classic, Intel had already redesigned the chip internally remedied the known issue of pipeline stalling. The result was an improvement of 30% in the execution of instructions. The frequency quickly sped up, reaching the GHz and half way through the year 2000, Intel launched a 1.13 GHz version, but a popular hardware site demonstrated in a review, that this was not actually stable enough to run the Linux kernel. The problem was found in the integrated cache which could not be taken to speeds over 1 GHz. It took Intel 6 months to solve the problem and finally release 1.1 and 1.13 GHz versions in 2001.

Tualatin

The third version, Tualatin, was actually little more than a test for the new production process at 130 nm, and had a good performance, especially the L2 512 Kb cache version, called Pentium III-S. This version was optimal for servers, in particular in situations where consumer-care was crucial. These processors were released during 2001 until 2002 at speeds of 1.0, 1.3, 1.2, 1.26, 1.33 and 1.4 GHz. Intel did not want a repeat of the situation by which Celeron had been able to compete with the "big brother" Pentium II, so Tualatin was not produced at speeds higher than 1.4 GHz, the lowest frequency in the range of Pentium 4 speeds. Later, the Pentium M would have proved that P6 architecture was capable of speeds up to 1.7 GHz with production process at 130 nm.

The successor

The Pentium III was replaced with the Pentium 4 Willamette, released in November 2000. This new architecture, known as NetBurst, was decisively oriented towards an increase in clock speed, especially thanks to a long 20-stage pipeline. Also the BUS was innovative, still at 100 MHz, but quad pumped, in other words able to transmit 4 bits simultaneously. This is the reason it is traditionally indicated as being at 400 MHz.

■ **Figure 2.345**
Katmai and Coppermine processors.

1999 d.c.

- The problem of the **millennium bug** becomes imminent. Many computer systems only handle the last two digits of the year, for example abbreviating 1978 in 78, so the passage to 2000, interpreted as 00 would cause considerable problems in calculating lengths of time, because these systems believe 00 as previous to 99. In the previous months this problem had worked up a general state of uneasiness, because it is a risk for any system which keeps track of the date. The fear is of malfunctioning in areas like for instance surveillance systems, air traffic control, nuclear installations, satellite navigation control systems and so on… In the end, thanks to the enormous debugging work carried out in years and months gone by, the passage from 1999 to 2000 happens without severe problems, although a few payment systems and train traffic management seize up. It is estimated that the amount of money spent in solving the Millennium Bug is around 400 billion dollars.

1999 d.c.

- **Bunny**, a short film by **Blue Sky**, wins the Academy Award for the best animation film. It is the first short film to use Global Illumination algorithms for its whole length. For the rendering, 41 Alphaserver 4100 IBM Racks were needed, each one with 4 Alpha processors at 533 MHz, 4 MB cache, 2 GB of RAM and a 4.3 GB Hard Disk.

■ **Figure 2.346**
Blue Sky Studios used its own software for rendering, with porting on the Compaq Tru64 UNIX system.

1999 d.c.

- **Pixar** produces its second long film, **Toy Story 2**.

■ **Figure 2.347**
The DVD cover, and the "baddie" of the film.

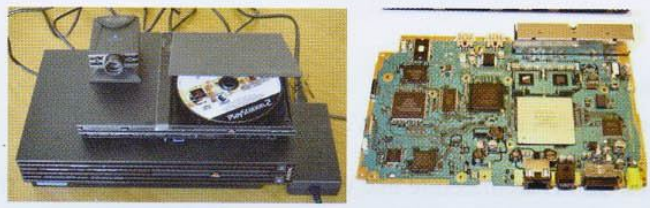
1999 d.c.

- In order to create the character Imhotep of the film **The Mummy, ILM** use a combination of models and shifting maps, which punch holes in the geometry (Implicit surface).

■ **Figure 2.348**
In order to animate the mummy, which is completely made in CG, the team uses a Vicon 8 motion capture system and animations with key frames; procedural animation manages the internal organs and falling bits of rotten skin.

2000 d.c.

- The **Playstation 2** is launched on the market. Following the footsteps of the first Sony console, the PS2 soon becomes the most popular games machine, selling over 100 million units. A mini spin-off (Playstation 2 Slim) is born for this machine, as it had been for the Playstation. It is smaller and the ethernet connection is incorporated.

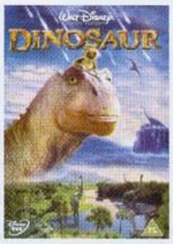


■ **Figure 2.349**
Unlike the Xbox console, online gaming is free (without a contract), using the PS2Online, even though software houses can use their own server to make games run online.

2000 d.c.

- **Disney** produces the film **Dinosaur**. The insertion of a whole cast of 3D animated characters, in a world of real life backgrounds, had never been carried out before, until Disney Feature Animation created an animation department for the production of this type of films.

■ **Figure 2.350**
In order to give life to dinosaurs and lemurs, 15 computer engineers wrote 450 programmes; 120 of these are plug-ins for Maya and Softimag/3D.



2000 d.c.

- **Windows ME**, Millennium edition, is a 32 bit operating system released on 14th September 2000 by Microsoft. This operating system, destined for home use, was developed starting from Windows 98 and mainly contains some updates such as Internet Explorer 5.5. One of the better improvements is the passage to Windows Media Player 7, the major rival of Real Player, the dominating multimedia player. In any case, both Windows Media Player and Internet Explorer were downloadable from Internet. Windows ME is a criticised operating system because of its tendency to crash frequently and its poorly detailed graphics.

A completely new program it introduced was Movie Maker which enables video editing, projected to be easy to use. The most significant change is the removal of the option of rebooting the PC into MS-DOS mode and the introduction of a backup and restore system for the system registry. In theory, this should represent a step forward: users no longer need to know DOS commands for system maintenance and problem-solving. In practice, the absence of DOS is a relevant obstacle for system management and the registry setup restore causes big problems: performance, which has never been one of Windows' strong points, is noticeably reduced; it is not a stable enough system to face all the common problems that may occur. Moreover, the system restores settings at every reboot, making it extremely difficult for inexperienced users to apply necessary or desired changes, like for instance removing viruses or undesired programs.

■ **Figure 2.351**
Many users were not at all impressed with this operating system, some sustained it deserved to be no more than an update of Windows 98 instead of an actual new version, others instead defined it as the worst Windows version since 3.0.

2000 d.c.

- Throughout the year, in Europe, **15 billion** SMS have been sent. The number of GSM users in the world is over **half a billion**. In other words, **1 human being out of 12 has a GSM cellular phone**.

2001 d.c.

- The **Xbox** is a console produced by **Microsoft,** sold from 15th November 2001. It is Microsoft's first attempt to enter the world of game consoles.

Microsoft crafted the Xbox with components which were altogether similar to those in a standard PC, which is usually not the case in the production of game consoles, which invariably adopt an engineering method with self-made architecture. The Xbox is made up of:

- A Intel Celeron (Coppermine) CPU with 733 MHz clock speed.
- A graphic processor nVIDIA NV2A which compared to the video card of a normal PC is somewhere between GeForce 3 and 4.
- 64 MB of RAM.
- DVD-ROM unit.
- ATA hard disk of 8 - 10 GB.
- Ethernet connection for LAN at 10/100 Mbit/.

Although the Xbox was based on PC architecture and executed a reduced version of the kernel of Windows 2000, a few restrictions have been introduced to avoid use with applications which are not approved by Microsoft.

It did not take long for the hacker community to find away to work round the limitations, managing to install an adapted version of GNU/Linux making the Xbox usable just like any PC without hindering its functions as a games console.

■ **Figure 2.352**
The Xbox was criticised for its awkwardness and size. This is mainly due to the use of a standard-size hard disk and a DVD-ROM reader. It is in any case more compact than an equivalent desk PC with similar hardware.

2001 d.c.

- **Final Fantasy**: it is the first photorealistic animation film, completely created in CGI.

■ **Figure 2.353**
Scenes from the film "Final fantasy".

2001 d.c.

- Microsoft officially releases **Windows XP**, the new version of its operating system. Its code name is Whistler, and is an operating system produced by Microsoft based on Windows NT architecture. It is the next-to-last version of Windows or client computers. The name "XP" comes from the word eXPerience. Microsoft launched two very similar versions at first: Home and Professional. The first had been thought out for domestic use and the second was more expensive and aimed at company utilization, and had some extra features, such as support for dual -processor systems. In November 2002 two hardware-specific versions of XP were announced:

- Windows XP Media Center Edition, for home theatre systems based on PCs. This version is sold pre-installed in the computer and can not be bought separately in shops.
- Windows XP Tablet PC Edition, projected for notepads/laptops with a special touch-screen facility, and writing-recognition functions.
- Windows XP x64 edition, conceived for computers with 64 bit processors.
- Windows XP **Starter Edition**. It is a low cost version of Windows XP available in Asia and South America. It is similar to Windows XP Home but some features have been removed or disabled by default.

■ **Figure 2.354**
Opening logo for Windows XP and initial screen.

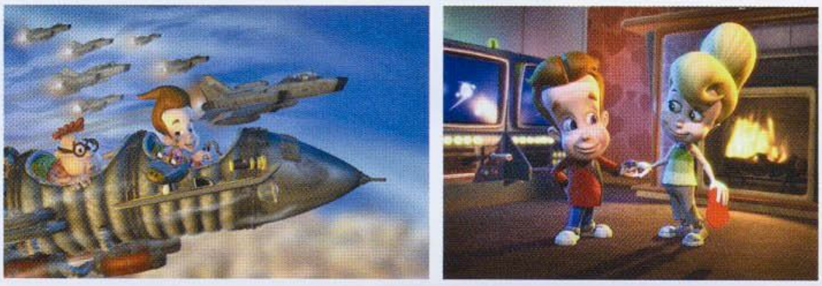
2001 d.c.

- **Mac OS X** is born, operating system for Apple Computer, installed on Macintosh machines. Mac OS X is not just version number 10 of the popular Mac OS, born in 1984 with the first Macintosh computer: it has been completely rewritten and is, to all intents and purposes, a new operating system.

Mac OS X was created combining Darwin, an Open source derived from UNIX and XNU kernels and from a kernel based on the Mach microkernel with a graphic interface (GUI) called Aqua, developed by Apple Computer.

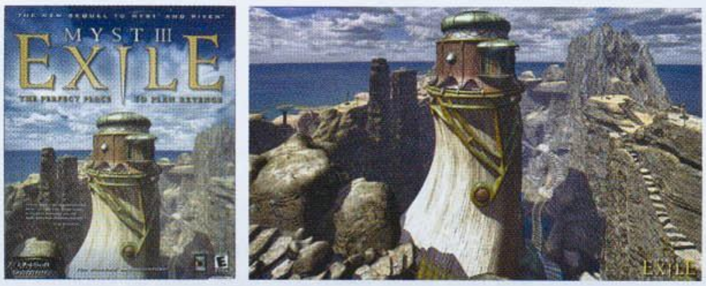
■ **Figure 2.355**
This operating system had its first commercial release in 2001. It is implemented on IBM G3, Motorola G4 and IBM G5.

2001 d.c.
- **Jimmy Neutron: Boy Genius** is the first quality animation film made with consumer-level hardware and software. Lightwave was used for modelling, illumination, effects and all of the animation except for the characters and some scenes with crowds. Project: Messiah was used for setup and animating the characters, Photoshop for creating the textures, Magpie for lip movement synchronization and Maya fusion for compositing.



■ Figure 2.356
Images from the film "Jimmy Neutron: Boy Genius".

2001 d.c.
- **Prestostudios** release **Myst III: Exile**, which sinks players into territories with beautiful textures. The previous versions used a slide-show style for visualizing the scenes, instead Myst III allows one to have a 360 degree view of every scene.



■ Figure 2.357
DVD cover and images from one of the beginning locations.

2002 d.c.
- In the film **Lord of the Rings: The Two Towers** enormous crowds with artificial intelligence are created by means of the commercial software, **Massive**.
This revolutionary software, formulated and created by Stephen Regelous of WETA Digital, animated the astonishing Prologue of The Fellowship of The Ring and was used repeatedly for The Two Towers. Regelous created the program with a spirit of devotion towards the technology of artificial intelligence. "I concocted it using an approach inspired by artificial life, rather than by what would have been done for a typical system." He explains. Massive creates "agents", with their own parameters of randomness and the ability to make decisions in a crowded situation, with many other creatures moving around near them. "In order for these agents to react naturally to what goes on around them, they must have the same senses that we have. They can see, hear, they can feel contact when they collide, and they can perceive what is around them".
Every agent also has his own personality attributes, like audacity, aggressiveness, cowardice, and so on. "and then there are parameters which measure how dirty, tall, or tired they are. So there are many ways for each of these agents to be able to act and take on a unique entity." says Regelous.
But Jackson does not see these Massive agents as completely animated creatures. "They simply crowd up into armies and then we press a button and they go off to fight on their own."
"Each one of these individuals has an Artificial Intelligence brain" explains the technical director of Massive, Geoff Tobin. "A part of this brain decides what to do on the basis of what he is dong in that moment and what he is able to do; the other part manages the data on all this, telling him all the information about the type of terrain beneath his feet, about the nearby enemies and allies, about navigation. In a certain sense they are not all that different to live walk-on parts".

■ Figure 2.358
A frame from "Lord of the Rings" and a screenshot from the Massive software program.

2002 d.c.

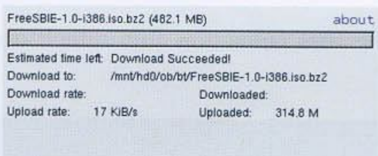
- A supercomputer at 52.4 teraflops is announced: it is the **Cray X1** model. It has processors with a power of 12.8 gigaflops and is capable of hosting up to 4,096 CPU units. The **FLOPS** (**Fl**oating-point **O**perations **P**er **S**econd) represent the number of floating-point calculations with fractional numbers carried out in one second by a processor. It is used to measure and compare the speed and performance of a processor (CPU or FPU). Its multiples are mFLOPS (megaFLOPS) equivalent to a thousand billion operations per second. The following chart compares some of the systems developed by different computer companies in the last century.

COMPUTER	*mFLOPS*	*Gigaflops*	*YEAR*
IBM Gene/L	*280,000,000*	*280,000*	*2006*
Cray X1	*52,000,000*	*52,000*	*2002*
PS3	*218,000*	*218*	*2006*
7800 GTX	*200,000*	*200*	*2006*
Cray Y-MP C90	*16,000*	*16*	*1991*
Mac G4	*15,000*	*15*	*2004*
Itanium 733	*2,022*	*2.2*	*2000*
CRAY 2	*1,700*	*1.7*	*1985*
P4-1700SSE	*1,332*	*1.3*	*2001*
CRAY-MP	*420*	*0.420*	*1982*
G3-700	*311*	*0.311*	*1998*
CRAY-1	*166*	*0.166*	*1976*
CDC 6600	*9*	*0.009*	*1964*
ENIAC	*0.005*	*5000 op/sec*	*1946*
MARK 1	*0.000001*	*1 op/sec*	*1937*

Twenty seven years after his first computer, Cray makes a comeback with an incredibly innovative system destined to become one of the most powerful computers ever. The Cray X1 is able to reach a power peak of 52.4 thousand billion floating-point operations per second (teraflops).
It can address up to 65.5 terabytes of memory. The price: 2.5 million dollars… in its default configuration! The X1 is dedicated to utilizations where there is a necessity for extensive volumes of calculations, such as defence structures, research centres, meteorology institutes and firms in car industry, aerospace industry, chemical and pharmaceutical industries.

2002 d.c.

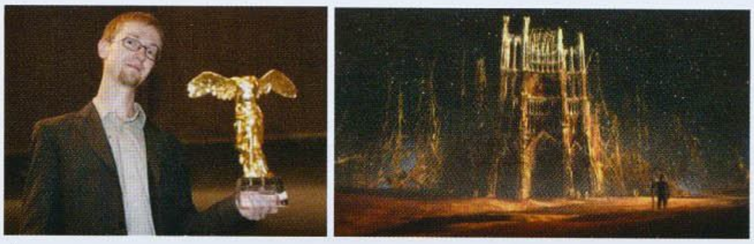
- The first version of the peer-to-peer program, **Bit Torrent**, is published online, written by **Bram Cohen** with the sponsorship of John Gilmore, ex Sun. Its original feature is the fact that it avoids the bottleneck which all previous peer-to-peer systems had endured, by transforming each user into a downloader and at the same time into an uploader, whereas before most users would download only, without contributing to the exchange process. With this method, the higher the user demand is for a certain file, the higher the number of users offering it becomes, and so downloading becomes much faster. With the old system, if a file was required by a large number of users, there would be a paralysis in the download procedure.



■ **Figure 2.359**
Screenshot of the first Torrent Client.

2002 d.c.

- **Tomek Baginsky** of Platige Image creates **The Cathedral**, a short animation film with a gothic style to it. This will win it the prize as best short animation film at the SIGGRAPH of that year. Work on it began in 1999 and has been estimated that in the two and a half years of production, Baginsky worked on the project for roughly 15 months. It was wholly achieved with 3Ds Max with the aid of Brazil as a rendering engine. Other software was of assistance as well, such as Stitch for the simulation of clothing, which was integrated starting from 3Ds Max 8.



■ **Figure 2.360**
According to the author, the most gratifying part was the creation of the painting elements. The finest parts of the short were, in Tomek's opinion, the creation of matte paintings and the union of 3D environments with 2D backgrounds.

2003 d.c.

- A one billion dollar investment, 4 years work and 1,200 engineers. This is what it took **IBM** to get to the presentation of what the society claimed to be "the most advanced mainframe in the world": the new **z990 server**, better known with its codename T.Rex.

The z990 server permits dynamic distribution of workloads within the machine on the basis of priorities established by the client. The system supports the most widespread standards: Linux, WSDL, SOAP and XML.

■ **Figure 2.361**
The CPU module comes in the form of a MultiChipModule, as large as a floppy disk and inside it incorporates 16 chips for a total of 3.2 billion transistors.

2003 d.c.

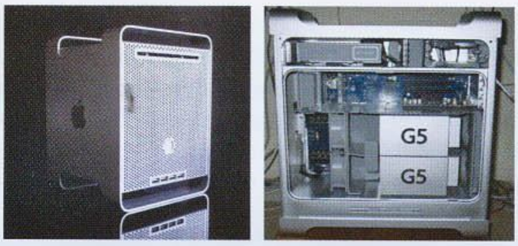
- In the film **Matrix Revolution**, the most detailed and realistic CGI animation ever seen is fulfilled, in the scene of the "super punch".

■ **Figure 2.362**
Final frames and sequence of images of facial deformation.

2003 d.c.

- **Apple** introduces **Power Mac G5**. Power Mac G5 is the name Apple Computer gave to the model of Power Macintosh which uses a PowerPC G5 processor.

■ **Figure 2.363**
The PowerMac G5 series is made up of three models, all dual-processor endowed. The models are dual at 1.8 GHz, dual at 2.0 GHz and dual at 2.5 Ghz.



2004 d.c.

- **The Polar Express** is the first animation film which uses motion capture intensively for movements and facial expressions.

■ **Figure 2.364**
Two frames of the Sony produced film.

2005 d.c.

- **Autodesk purchases Alias**. Many suspect an interruption in the development of Maya, whereas others expect a combination of the two software products, 3Ds Max and Maya, in one software. Others still foretell the slow demise of both of them in the wake of new software, completely rewritten. Three years later, still nothing has happened.

2005 d.c.

- **Blue Gene** is the name of a type of architecture planned by **IBM** to form the new generation of supercomputers, developed to work with a level of calculation power which goes from tens of Teraflops and reaches Petaflops. The first Blue Gene system, the **Blue Gene/L** was developed in collaboration with the Lawrence Livermore National Laboratory (LLNL). The system develops a theoretical peak power of 360 Teraflops and, according to the Linpack test, generates 280 Teraflops, through 65,536 calculation nodes distributed in 64 towers.

Each calculation or communication node is a single ASIC with an associated DRAM memory. The ASIC incorporates two PowerPC 440 processors at 700 MHz built with a technology of 130 nanometres. Each processor is provided with two floating-point calculation units with double precision and a subsystem which handles the cache and DRAM memory as well as managing inter-process communication. The two FPU units provide each Blue Gene node with a theoretical power of 5.6 Gigaflops.
Assimilating all the subsystems in one chip helps lower the dispersion of power. In fact, each chip consumes only 17 Watts, the DRAM included. This allows one to assemble a vast number of chips in a small space and consequently to insert 1024 calculation nodes and their attached communication elements in a single tower, whilst maintaining reasonable consumption and power dissipation. If one analyses the system comparing Flops per Watt or in Flops per Watt by the square metre, one finds that the Blue Gene project is an extremely efficient one.

IBM intends to create Blue Gene/P in 2006. Blue Gene/P should be the first supercomputer to go beyond the threshold of Petaflops, that is one million billion floating-point operations per second.

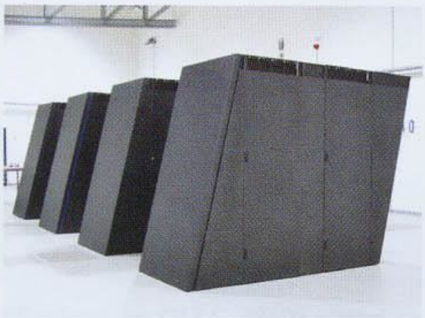
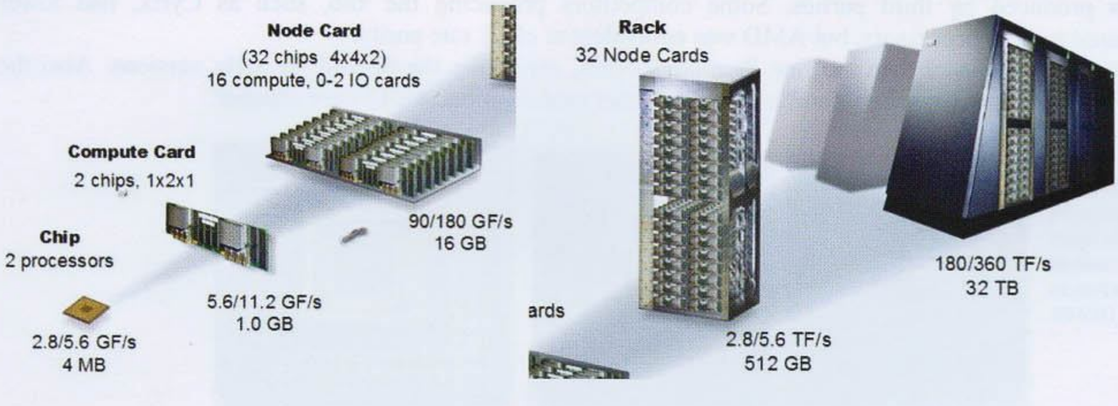
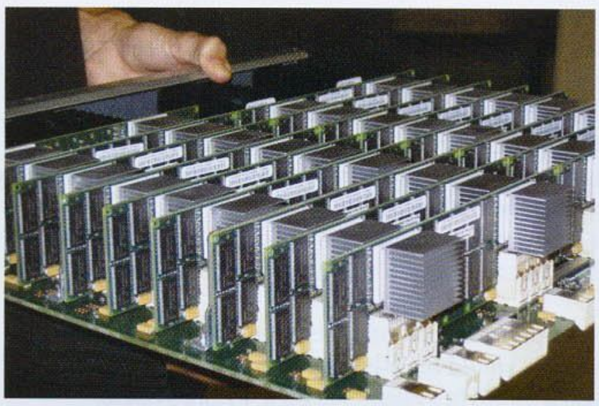


Figure 2.365
- Assembly and storage place for Blue Gene/L at the Lawrence Livermore National Laboratory.
- Construction scheme of the supercomputer.





2005 d.c.
- **AMD** releases its first **Dual-core** processors.

1991

The **Am386** is an x86 microprocessor produced by **AMD**. With 1 million copies sold by AMD, this clone, 100% compatible with the Intel 80386 gave the company a position of full competition rather than being a licensed producer.
Although the processor was ready to be released before 1991, Intel held it still with a court case: AMD had previously been a producer of Intel-designed chips, with a license from Intel itself, and on request; AMD interpreted the contract to be valid for all Intel processors, whereas Intel interpreted it to be valid only for those preceding the 80386 (excluded). After a few years in court, AMD won the case and distributed its Am386. This event severed the monopoly of Intel and gave computer prices a downward shove.
Whereas Intel CPUs were clocked at 33 MHz, AMD released versions 386SX and 386DX at 40 MHz, giving their architecture a longer lifespan. The AMD 386DX-40 was very popular amongst small producers of clone PCs and computer-lovers who had their eye on their wallet, because the 386DX-40 could reach or even overtake the 486SX-25 in many a benchmark, in spite of its lower cost, and it had peaks which were often higher even than the 486DX-33 in many practical applications. This was mostly due to an FSB going at full speed, whereas the fastest 486 at the time did not have a bus that went beyond 33 MHz. The floating-point calculation performance could be improved cheaply with a 387DX mathematical co-processor, but in any case could not get anywhere near the 486DX, making the 386DX a bad choice for 3D applications or CAD.

■ **Figure 2.366**
An Am386DX-40 processor.

1993

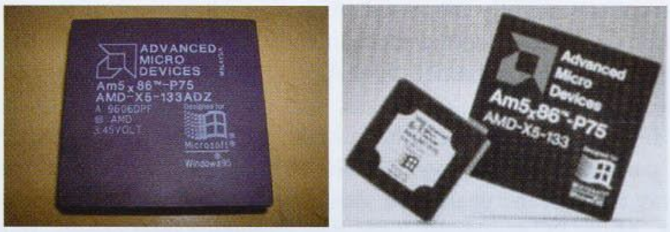
The **Am486** was an x86 microprocessor produced by AMD based on the design of Intel's 80486. AMD released a 40 MHz version at the same price as Intel's 33 MHz one. The first Am486s were simple imitations of the Intel processor, but soon started working at 3.3 V, against 5 V of Intel processors, making immediate upgrades impossible until the arrival of adapters produced by third parties. Some competitors producing the x86, such as Cyrix, had lower performance compared to Intel processors, but AMD was equivalent at clock rate parity.
The faster models were even faster than the first Pentium models, especially the 60 and 66 MHz versions. Also the equivalent Intel 80486DX4s were expensive and required a socket modification, AMD were cheaper.

■ **Figure 2.367**
The DX4 had a double the cache of an Am486, and this granted it performance with peaks higher than those of AMD, however AMD's DX4-100 cost less than an Intel DX2-66.

1995

The **Am5x86** is a x86 microprocessor produced by AMD to be able to carry out direct upgrades on systems based on 486 processors; with this aim it was one of the fastest and most compatible processors. It is a standard 486 processor with an internal multiplier set to 4, allowing the CPU to work at 133 MHz on systems without official support for DX2 or DX4 processors. Like most of the last 486 processors, it would support 16 Kb of L1 write-back cache memory, and not 8 as in most cases. A rare 150 MHz version also was produced. As a clock multiplier set to 4 is not compatible with Socket 3, AMD made the processor interpret motherboards set to 2 as if they were set to 4. The chip was also compatible with Socket 1 and 2 of the older 486 models, but in this case a voltage adapter was needed. The combination of extremely high frequencies and write-back memory support allowed the 5x86 to reach or even surpass a 75 MHz Intel Pentium in professional applications.
The CPU is often overclocked to 160 MHz, sometimes reaching the performance of a Pentium-100. Although it has been rumoured that some were able to push the chip up to 200 MHz, this would have been difficult to obtain because of the scarce number of video cards that could have been combined with a 50 MHz system bus on a VESA system.

The 5x86 is also noteworthy for having been the first CPU to be evaluated with a controversial PR rating. As it was equivalent, in the benchmarks, to a Pentium-75, AMD named the last of these processors "Am5x86-P75" models. The sales of these models gave the firm a breath of air while it was having problems with the production of the AMD K5. AMD produced the 5x86 for PCs until 1999.

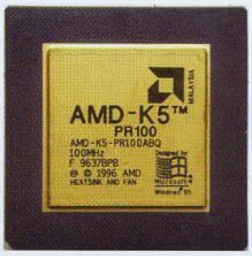


■ **Figure 2.368**
AMD 5x86-P75 processor for PGA Socket.

1995

The **AMD K5** is a x86 compatible microprocessor developed by AMD introduced one year late, because of problems reaching the prefixed clock speed, in order to compete directly with the Intel Pentium. None of the versions had MMX instructions support, although the processor was extremely advanced and more similar to a Pentium Pro than to a simple Pentium: it was entirely based on RISC architecture, with a decoding unit for x86 instructions. This processor represented AMD's chance to overcome Intel technologically, but production problems connected to low clock speed weighed upon the fine design of the CPU, which had a performance in floating-point calculations lower than the Pentium. Because of its late arrival on the market and its slightly unsatisfactory performance, it was not as well greeted commercially as its precursor, the Am486, or its successor, the K6.

Processor	Clock Speed	Bus Speed	Clock Muiltiplier
K5 PR 75	75 MHz	50 MHz	1.5x
K5 PR 90	90 MHz	60 MHz	1.5x
K5 PR 100	100 MHz	66 MHz	1.5x
K5 PR 120	90 MHz	60 MHz	1.5x
K5 PR 133	100 MHz	66 MHz	1.5x
K5 PR 150	116.5 MHz	60 MHz	1.75x
K5 PR 166	116.5 MHz	66 MHz	1.75x



■ **Figure 2.369**
A typological chart of AMD K5 processors. A K5 PR 100 model.

1997

The **AMD K6** is a x86 compatible microprocessor developed by AMD. This processor has been conceived in particular to be compatible with already existing systems based on the Intel Pentium, which it was able to replace with equivalent performance and at a considerably cheaper price. This guaranteed the CPU a strong impact on the market and the chance of competing directly with Intel. The design of the K6 was based on the Nx686, which NexGen was developing when it was englobed by AMD. NexGen intended to produce the processor with its own socket format but AMD imposed the Socket 7 standard, in order to have greater compatibility, it added MMX instructions support and called the processor K6. Although the initials might lead one to think that there was a certain continuity from the K5, the design was actually completely different, created by the firm that had been englobed by AMD. The K6 was launched at speeds of 166 MHz and 200 MHz in April 1997 and the same year a 233 MHz version came out. The release of a 266 MHz version was not possible until spring 1998, when AMD was able to pass onto the 0.25 micron production procedure. The last modification was applied in May with a 300 MHz version before the introduction of the K6-2 and the K6-III. AMD initially used a "PR2 rating" (Pentium II rating) to define its processors, a sort of updated PR rating. This was interrupted, due to the fact that in this way the frequencies indicated with the rating were very close to the real ones.

■ **Figure 2.370**
The three versions of the K6. The K6, the K6-2 and the K6-III.

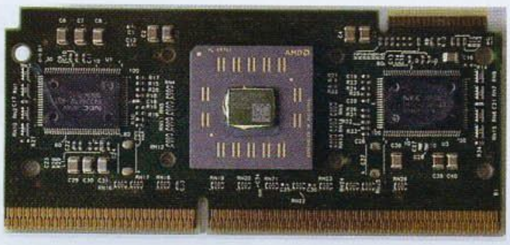
1999

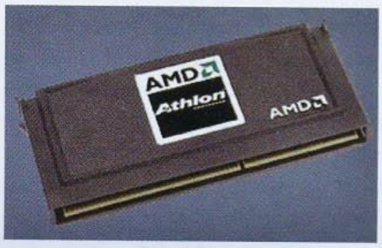
Athlon is the name given to a range of CPUs projected and produced by AMD. The Athlon, or **Athlon Classic**, was the first processor of the 7th generation of AMD x86 processors called K7, in honour of the previous K6 series, although they were completely different in terms of construction in many ways. AMD continued to call the 8th generation of processors Athlon too.

The original Athlon came out on 21st August 1999 and the name Athlon was chosen by AMD as a diminutive for "decathlon", so that it could gain more visibility and distinguish itself from the names of its old models which had not had much success. The first version of the Athlon core, called K7 or Argon, became available with an initial speed between 500 and 700 MHz for Slot A. Later the argon core was revisited and renamed K75 (respectively Pluto and Orion cores) which could reach the threshold of 1000 MHz thanks to the new production process (0.18 microns instead of 0.25) and with operative voltage slightly increased. Also this adjustment of the processor was compatible with the standard x86 set of instructions and was inserted mechanically into a slot which was similar (but not compatible with the pins) to Slot 1 in Pentium II processors, namely Slot A.

From the commercial point of view, the Athlon Classic was an enormous success, not only because of its deserving qualities, but also because Intel, which was usually reliable, had encountered production, design and quality control problems in that period. In particular, the passage of Intel to the 0,18 micron production process, which began in late 1999 and lasted until half way through the year 2000, was disorganized and the result was a critical shortage of Pentium III processors on the market. Many PC producers, until then exclusive Intel clients, found the Athlon to be a convenient combination of excellent performance and reasonable cost and were delighted to discover its abundant availability. Unlike Intel, AMD organized the change in production process in a fairly intelligent way. It could provide a great number of processors and Athlon increased its sales to a great extent.

■ **Figure 2.371**
Athlon processor for Slot-A. Notice the L2 cache, external to the chip.



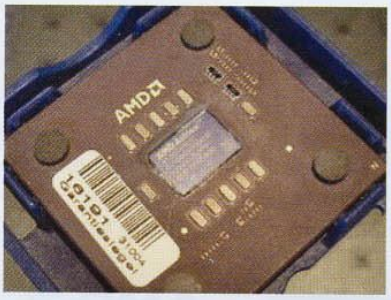


2000

The second generation of **Athlon** appears, **Thunderbird**. This version of the Athlon came out in a PGA format and was no longer a card format for slot A. Of course, a socket to host the CPU was introduced simultaneously to its release, Socket 462, also called Socket A. In view of the demand, versions of this processor for slot A from 650 to 1000 MHz were released. The main difference with past models was the cache; much like Intel had done, replacing the old Pentium III Katmai with the much faster P-III Coppermine, AMD replaced the external low-speed 512 Kb cache with a 256 Kb full-speed cache, integrated on the chip. It may seem that a halved cache would lower performance, but in actual fact the high speed surpasses this problem easily and allowed a level of performance which was by far greater than the previous versions.

Thunderbird was the most successful processor by AMD since the Am386DX-40, ten years before. The design of motherboards had improved greatly in the meantime, and the small circle of PC producers supporting Athlon grew to the point of including all the major producers. The new factory in Dresden started working, allowing a greater production at lower costs and the quality of production was improved by passing on to copper connections.

■ **Figure 2.372**
Athlon Thunderbird.

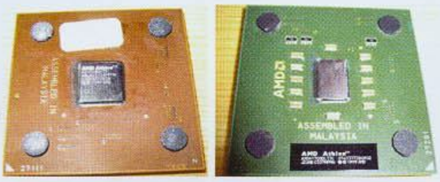


2001

The **Athlon XP** was a processor built in 2001 by AMD to replace the Athlon Classic and try to re-conquer the pinnacle of the market. In terms of performance, the Athlon core Thunderbird had easily obscured its rival, the Pentium III and the first Pentium 4 models to come out were late and had a very disappointing level of performance.

In April 2001 the situation had already changed, the P4 at 1.7 GHz gave the impression that Athlon Thunderbird was not going to be able to maintain performance leadership for much longer, and problems with heat and consumption in the design of the Thunderbird indicated that it was not going to be possible to take it above
1,400 MHz. In fact, even at that speed it would heat up too much. AMD released the first version of the Athlon XP, considered by many the third version of the Athlon Classic, with a new core called **Palomino**. This version was the first to support SSE, already integrated in the Pentium III, like for instance 3Dnow! Professional by AMD itself. It was initially sold at speeds from 1333 to 1733 MHz.

The Athlon XP was marketed with a PR rating system, which compared its performance with that of the Athlon Thunderbird. Because the IPC (instructions per clock cycle) of the Athlon XP were much greater than the Pentium 4 (and about 10% higher than the Thunderbird), it was much more efficient and gave equal results at a lower speed and higher performance at equal speed, so a purely advertisement-oriented clash on the number of GHz (which Intel was stressing at the time), would have been lost to the competitor, even if the Athlon was actually faster.
The fourth generation of Athlons, the core **Thoroughbred**, was released on June 10th 2002 with a speed of 1.8 GHz or 2200+ according to the PR rating system. Two new Athlon XP models, the 2400+ (2000 MHz) and the 2600+ (2083 MHz or 2133 MHz, depending on the speed, 100/133 MHz on the Front Side Bus) were announced on August 21st and launched on the market. Also 2700+ and 2800+ versions of the Thoroughbred core were produced but distributed in insignificant quantities because of production problems and because of the arrival of the new core, Barton. The design of the "Thoroughbred" did not differ at all, except that it was built with a production process of 0.13 microns, as opposed to the 0.18 microns of the Palomino.
The fifth generation of this processor, the **Barton** core, was released in early 2003 with PR ratings of 2500+, 2600+, 2800+, 3000+ and 3200+. Although it was not faster than the Thoroughbred in terms of Megahertz, it earned itself the highest PR rating at equal speed, including 256Kb extra of full-speed L2 cache integrated on the chip, and a faster FSB. The Thorton core was a spin-off of the Barton with half of the L2 cache disabled.



■ **Figure 2.373**
- The logo of the AMD Athlon XP.
- Athlon XP 1700+ Palomino.
- Athlon XP 1700+ Thoroughbred.
- Chart with Athlon XP processors.

Cpu	Clock effettivo	Molt.	Freq. di bus	Core	Cache L1	Cache L2	Cache L3	Contr. memoria	64bit	NXbit	Socket	Micron
Athlon XP 3.200+	2,2GHz	11x	400MHz	Barton	128K	512K	no	esterno	no	no	A	0.13
Athlon XP 3.000+	2,1GHz	10.5x	400MHz	Barton	128K	512K	no	esterno	no	no	A	0.13
Athlon XP 3.000+	2,167GHz	13x	333MHz	Barton	128K	512K	no	esterno	no	no	A	0.13
Athlon XP 2.800+	2,083GHz	12,5x	333MHz	Barton	128K	512K	no	esterno	no	no	A	0.13
Athlon XP 2.800+	2,25GHz	13,5x	333MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.700+	2,166GHz	13x	333MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.600+	2,083GHz	12,5x	333MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.600+	2,133GHz	16x	266MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.600+	1,917GHz	11,5x	333MHz	Barton	128K	512K	no	esterno	no	no	A	0.13
Athlon XP 2.500+	1,833GHz	11x	333MHz	Barton	128K	512K	no	esterno	no	no	A	0.13
Athlon XP 2.400+	2GHz	15x	266MHz	Thorton	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.400+	2GHz	15x	266MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.200+	1,8GHz	13.5x	266MHz	Thorton	128K	256K	no	esterno	no	no	A	0.13

Cpu	Clock effettivo	Molt.	Freq. di bus	Core	Cache L1	Cache L2	Cache L3	Contr. memoria	64bit	NXbit	Socket	Micron
Athlon XP 2.200+	1,8GHz	13.5x	266MHz	Thor	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.100+	1,733GHz	13x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18
Athlon XP 2.000+	1,667GHz	12.5x	266MHz	Thorton	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.000+	1,667GHz	12.5x	266MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 2.000+	1,667GHz	12.5x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18
Athlon XP 1.900+	1,6GHz	12x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18
Athlon XP 1.800+	1,533GHz	11.5x	266MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 1.800+	1,533GHz	11.5x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18
Athlon XP 1.700+	1,467GHz	11x	266MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 1.700+	1,467GHz	11x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18
Athlon XP 1.600+	1,4GHz	10.5x	266MHz	Thor-B	128K	256K	no	esterno	no	no	A	0.13
Athlon XP 1.600+	1,4GHz	10.5x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18
Athlon XP 1.500+	1,333GHz	10x	266MHz	Palomino	128K	256K	no	esterno	no	no	A	0.18

2003

The **AMD Opteron** was the first processor of the K8 series (Athlon 64, Athlon X2, Athlon FX) produced by AMD and also the first processor compatible with 64 bit thanks to the implementation of AMD64 technology. Released for the first time on 22nd April 2003, it aimed at the server market in competition with Intel Xeon and Itanium. They are available in three distinct versions, according to the maximum quantity of processors which are simultaneously usable. Apart from this element, the CPUs with the same initials are identical as far as architecture features are concerned: for example, the Opteron 144, 244 and 844 CPU models all have the same cache and clock frequency, and the only difference is the maximum number of processors which can be used at the same time (respectively 1, 2 and up to 8).

■ **Figure 2.374**
- The AMD Athlon Opteron logo.
- An Opteron processor.

2003

The **Athlon 64** (code names are "Clawhammer", "Newcastle", "Winchester", "Venice" and "San Diego"), produced by AMD, was the second desktop processor of the x86 family with 64 bit support (other 64 bit processors had been released before by Sun, Digital, Mips and IBM but were reserved for the workstation market).

This processor belongs to a series where three variations have been produced: The Athlon 64, Athlon 64 X2 (code names "Manchester", "Toledo" and "Windsor") and the Athlon 64 FX. All three, thanks to AMD64 technology, support 64,32 and 16 bits. The most important new implementation in the Athlon 64 is the integrated memory controller. This component is usually present in the Northbridge, and is used for letting the CPU communicate with the memory. The fact of installing a memory controller in the processor's die, means that signals no longer have to go through the FSB to get to the Northbridge and then go to the memory or vice versa, but instead there is a direct exchange with the memory. Also, this way the memory controller runs at the same frequency (clock) as the processor, gaining more performance. This implementation basically lowers memory latency (time it takes to respond) and generally makes increases in performance notable.
Another important piece of technology in the Athlon 64 is the Cool 'n' Quiet, derived from the previous Power Now! for mobile CPUs and similar to Intel's SpeedStep for mobile CPUs. Thanks to this asset, when the processor is not using its full power because it is only running a few background programs, the clock and voltage of the processor itself are lowered. This means reducing the electricity used and consequently the heat produced, passing through a TDP (Thermal Dissipation Envelope) declared by AMD to be of 89 Watts, to one of 32 or 22 Watts (lowering the processor clock in that order by 80 and 1000 MHz).
Athlon 64 CPUs are produced with a production process of 130 and 90 nanometres.

■ **Figure 2.375**
- An Athlon 64 3000+.
- Athlon 64 FX processor.
- Athlon 64 X2 processor.
- Chart with list of Athlon 64 and Athlon 64 FX processors.

Cpu	Clock effettivo	Molt.	Freq. di bus HT	Core	Cache L1	Cache L2	Cache L3	Contr. memoria	64bit	NXbit	Socket	Micron
Athlon 64 FX57	2,8GHz	14x	1GHz	San Diego	128K	1M	no	Dual DDR	si	si	939	0.09
Athlon 64 FX55	2,6GHz	13x	1GHz	San Diego	128K	1M	no	Dual DDR	si	si	939	0.09
Athlon 64 FX55	2,6GHz	13x	1GHz	SledgeHammer	128K	1M	no	Dual DDR	si	si	939	0.13
Athlon 64 FX53	2,4GHz	12x	1GHz	SledgeHammer	128K	1M	no	Dual DDR	si	si	939	0.13
Athlon 64 FX53	2,4GHz	12x	800MHz	SledgeHammer	128K	1M	no	Dual DDR	si	si	940	0.13
Athlon 64 FX51	2,2GHz	11x	800MHz	SledgeHammer	128K	1M	no	Dual DDR	si	si	940	0.13
Athlon 64 4.000+	2,4GHz	12x	1GHz	San Diego	128K	1M	no	Dual DDR	si	si	939	0.09
Athlon 64 4.000+	2,4GHz	12x	1GHz	SledgeHammer	128K	1M	no	Dual DDR	si	si	939	0.13

Cpu	Clock effettivo	Molt.	Freq. di bus HT	Core	Cache L1	Cache L2	Cache L3	Contr. memoria	64bit	NXbit	Socket	Micron
Athlon 64 3.800+	2,4GHz	12x	1GHz	Venice	128K	512K	no	Dual DDR	si	si	939	0.09
Athlon 64 3.800+	2,4GHz	12x	1GHz	Winchester	128K	512K	no	Dual DDR	si	si	939	0.09
Athlon 64 3.800+	2,4GHz	12x	1GHz	Newcastle	128K	512K	no	Dual DDR	si	si	939	0.13
Athlon 64 3.700+	2,2GHz	11x	1GHz	San Diego	128K	1M	no	Dual DDR	si	si	939	0.09
Athlon 64 3.500+	2,2GHz	11x	1GHz	Venice	128K	512K	no	Dual DDR	si	si	939	0.09
Athlon 64 3.500+	2,2GHz	11x	1GHz	Winchester	128K	512K	no	Dual DDR	si	si	939	0.09
Athlon 64 3.500+	2,2GHz	11x	1GHz	ClawHammer	128K	512K	no	Dual DDR	si	si	939	0.13
Athlon 64 3.500+	2,2GHz	11x	1GHz	Newcastle	128K	512K	no	Dual DDR	si	si	939	0.13

Cpu	Clock effettivo	Molt.	Freq. di bus HT	Core	Cache L1	Cache L2	Cache L3	Contr. memoria	64bit
Athlon 64 3.500+	2GHz	10x	1GHz	San Diego	128K	1M	no	Dual DDR	si
Athlon 64 3.200+	2GHz	10x	1GHz	Venice	128K	512K	no	Dual DDR	si
Athlon 64 3.200+	2GHz	10x	1GHz	Winchester	128K	512K	no	Dual DDR	si
Athlon 64 3.200+	2GHz	10x	1GHz	Newcastle	128K	512K	no	Dual DDR	si
Athlon 64 3.000+	1,8GHz	9x	1GHz	Venice	128K	512K	no	Dual DDR	si
Athlon 64 3.000+	1,8GHz	9x	1GHz	Winchester	128K	512K	no	Dual DDR	si
Athlon 64 3.000+	1,8GHz	9x	1GHz	Newcastle	128K	512K	no	Dual DDR	si

2005

Two years after the presentation of the first AMD Opteron, the Sunnyvale company presented the **Opteron Dual-Core** version of its server processor. The new dual core processors have integrated separated L1 and L2 caches within and a single memory control in common. The code names of the new processors are Denmark, Italy and Egypt. Opteron single core processors available until today are recognizable from the model number which increases by 2 for each one, reaching Opteron model 252. For the Dual Core versions the choice was made to increase model numbers by 5 each time. AMD has declared that these new processors, thanks to numerous optimizations, will have a 95 Watt TDP like the previous ones, in spite of the fact they host two cores in their die.
Each processor is marked with a model number type XYY, where the first digit (X) indicates what type of system the processor has been made for:

1- Single processor systems.
2- Double processor systems.
8- Systems with 4 to 8 processors.

The YY digits, instead, tell us the speed of the processor in relation to other models. From August 2nd 2005, the Sunnyvale company has announced that the CPU Opteron 1YY series will be produced with Socket 939 and not anymore with Socket 940; This will allow the production of motherboards to cost less and also the use of DDR unbuffered memory in place of the ECC Registered type which Opteron Socket 940 models require.

■ **Figure 2.376**
- The die of an Opteron Dual-core.
- An Opteron Dual-Core processor.
- A Tyan S4880 mainboard which allows the use of 4 opteron Dual-cores.
- Chart of available Opteron Dual-cores.

Sigla	Core	Frequenze Cpu	Moltiplicatore	Bus HyperTransport	cache	Socket	Processo costruttivo
140	SledgeHammer	1.4 Ghz	7x	800 MHz	L1=128 KB L2=1 MB	940	130 nm
142	SledgeHammer	1.6 Ghz	8x	800 MHz	L1=128 KB L2=1 MB	940	130 nm
144	SledgeHammer, Venus	1.8 Ghz	9x	800, 1000 MHz	L1=128 KB L2=1 MB	940, 939	130, 90 nm
146	SledgeHammer, Venus	2.0 Ghz	10x	800, 1000 MHz	L1=128 KB L2=1 MB	940, 939	130, 90 nm
148	SledgeHammer, Venus	2.2 Ghz	11x	800, 1000 MHz	L1=128 KB L2=1 MB	940, 939	130, 90 nm
150	SledgeHammer, Venus	2.4 Ghz	12x	800, 1000 MHz	L1=128 KB L2=1 MB	940, 939	130, 90 nm
152	Venus	2.6 Ghz	13x	1000 MHz	L1=128 KB L2=1 MB	939	90 nm
154	Venus	2.8 Ghz	14x	1000 MHz	L1=128 KB L2=1 MB	939	90 nm
165	Denmark	2x1.8 Ghz	9x	1000 MHz	L1=2x128 KB L2=2x1 MB	939	90 nm
170	Denmark	2x2.0 Ghz	10x	1000 MHz	L1=2x128 KB L2=2x1 MB	939	90 nm
175	Denmark	2x2.2 Ghz	11x	1000 MHz	L1=2x128 KB L2=2x1 MB	939	90 nm
180	Denmark	2x2.4 Ghz	12x	1000 MHz	L1=2x128 KB L2=2x1 MB	939	90 nm
240/840	SledgeHammer	1.4 Ghz	7x	800 MHz	L1=128 KB L2=1 MB	940	130 nm
242/842	SledgeHammer	1.6 Ghz	8x	800 MHz	L1=128 KB L2=1 MB	940	130 nm
244/844	SledgeHammer, Troy/Athens	1.8 Ghz	9x	800 MHz	L1=128 KB L2=1 MB	940	130, 90 nm
246/846	SledgeHammer, Troy/Athens	2.0 Ghz	10x	800 MHz	L1=128 KB L2=1 MB	940	130, 90 nm
248/848	SledgeHammer, Troy/Athens	2.2 Ghz	11x	800 MHz	L1=128 KB L2=1 MB	940	130, 90 nm
250/850	SledgeHammer, Troy/Athens	2.4 Ghz	12x	800 MHz	L1=128 KB L2=1 MB	940	130, 90 nm
252/852	Troy/Athens	2.6 Ghz	13x	1000 MHz	L1=128 KB L2=1 MB	940	90 nm
265/865	Italy/Egypt	2x1.8 Ghz	9x	1000 MHz	L1=2x128 KB L2=2x1 MB	940	90 nm
270/870	Italy/Egypt	2x2.0 Ghz	10x	1000 MHz	L1=2x128 KB L2=2x1 MB	940	90 nm
275/875	Italy/Egypt	2x2.2 Ghz	11x	1000 MHz	L1=2x128 KB L2=2x1 MB	940	90 nm
280/880	Italy/Egypt	2x2.4 Ghz	12x	1000 MHz	L1=2x128 KB L2=2x1 MB	940	90 nm

2005

Turion processors are available, destined to be used in portable computers because of the low electricity consumption, and the first Dual-core processors: the Opteron and the Athlon 64 X2 which give the Sunnyvale company a good advantage on the competing Intel, whose Dual-core processors for servers were expected only later, in 2006.
All last-generation processors are built on a 0.09 micron technology and gradually AMD is moving towards 65 nanometres (0.065 microns). The new Athlon 64 X2 CPU models are characterized by the adoption of Dual Core architecture, with 1 Mbyte or 512 Kbyte L2 caches specific for each core.

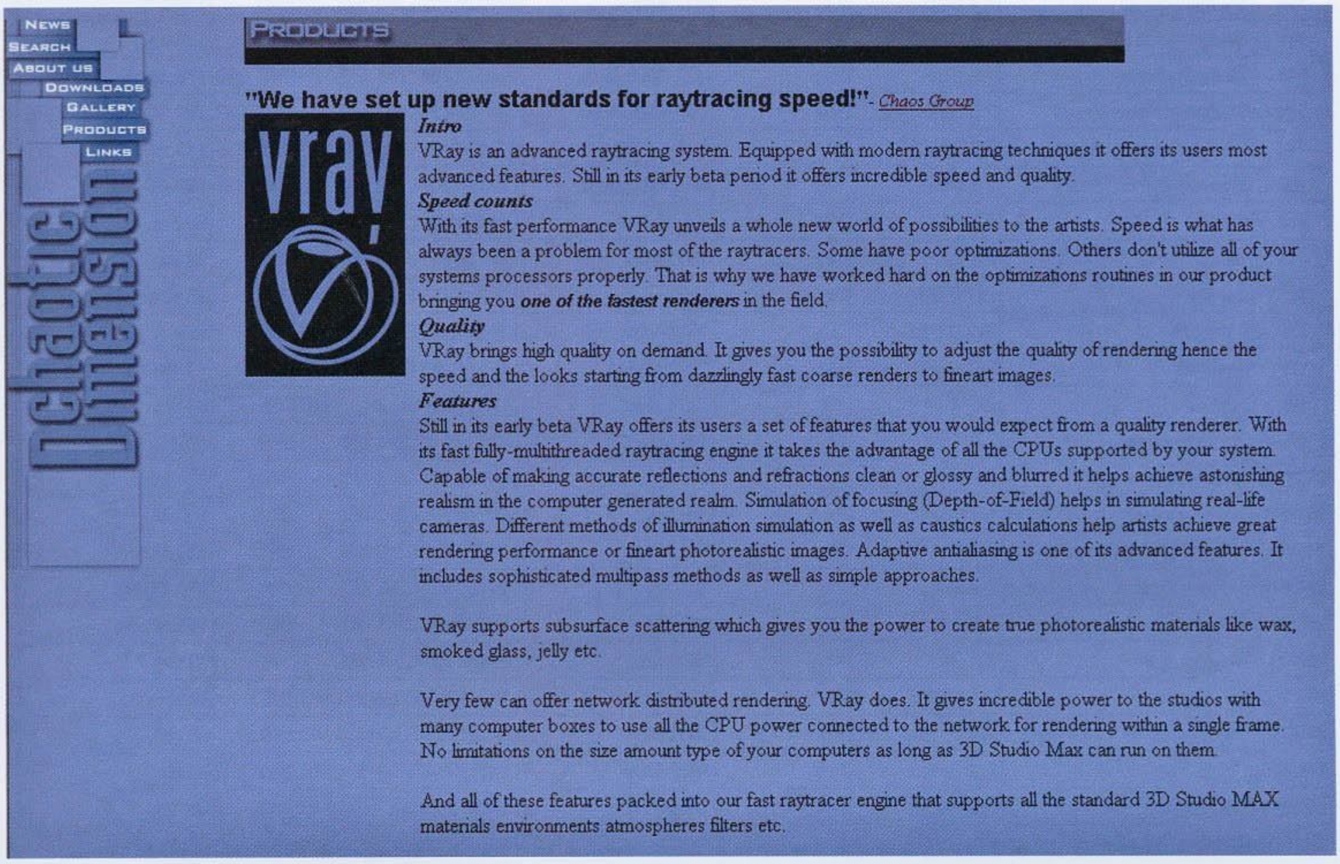
AMD 64 Athlon X2

Cpu	Clock effettivo	Molt.	Freq. di bus HT	Core	Cache L1	Cache L2	Cache L3	Contr. memoria	64bit	NXbit	Socket	Micron
Athlon 64 X2 4.800+	2,4GHz	12x	1GHz	Toledo	2x128K	2x1M	no	Dual DDR	si	si	939	0.09
Athlon 64 X2 4.600+	2,4GHz	12x	1GHz	Toledo	2x128K	2x512K	no	Dual DDR	si	si	939	0.09
Athlon 64 X2 4.400+	2,2GHz	11x	1GHz	Toledo	2x128K	2x1M	no	Dual DDR	si	si	939	0.09
Athlon 64 X2 4.200+	2,2GHz	11x	1GHz	Toledo	2x128K	2x512K	no	Dual DDR	si	si	939	0.09

■ **Figure 2.377**
- Logo of the AMD 64 X2 processor.
- Chart with available AMD 64 X2 processors.

2001

May: the **Chaotic Dimension site** opens. In *VRay* many features have not yet been implemented and others are disabled. A few days later the first images rendered with *VRay* are published and the initial information about the program will be given out.


News
Search
About us
Downloads
Gallery
Products
Links
Chaotic Dimension

PRODUCTS

"We have set up new standards for raytracing speed!"- Chaos Group

vray

Intro

VRay is an advanced raytracing system. Equipped with modern raytracing techniques it offers its users most advanced features. Still in its early beta period it offers incredible speed and quality.

Speed counts

With its fast performance VRay unveils a whole new world of possibilities to the artists. Speed is what has always been a problem for most of the raytracers. Some have poor optimizations. Others don't utilize all of your systems processors properly. That is why we have worked hard on the optimizations routines in our product bringing you ***one of the fastest renderers*** in the field.

Quality

VRay brings high quality on demand. It gives you the possibility to adjust the quality of rendering hence the speed and the looks starting from dazzlingly fast coarse renders to fineart images.

Features

Still in its early beta VRay offers its users a set of features that you would expect from a quality renderer. With its fast fully-multithreaded raytracing engine it takes the advantage of all the CPUs supported by your system. Capable of making accurate reflections and refractions clean or glossy and blurred it helps achieve astonishing realism in the computer generated realm. Simulation of focusing (Depth-of-Field) helps in simulating real-life cameras. Different methods of illumination simulation as well as caustics calculations help artists achieve great rendering performance or fineart photorealistic images. Adaptive antialiasing is one of its advanced features. It includes sophisticated multipass methods as well as simple approaches.

VRay supports subsurface scattering which gives you the power to create true photorealistic materials like wax, smoked glass, jelly etc.

Very few can offer network distributed rendering. VRay does. It gives incredible power to the studios with many computer boxes to use all the CPU power connected to the network for rendering within a single frame. No limitations on the size amount type of your computers as long as 3D Studio Max can run on them.

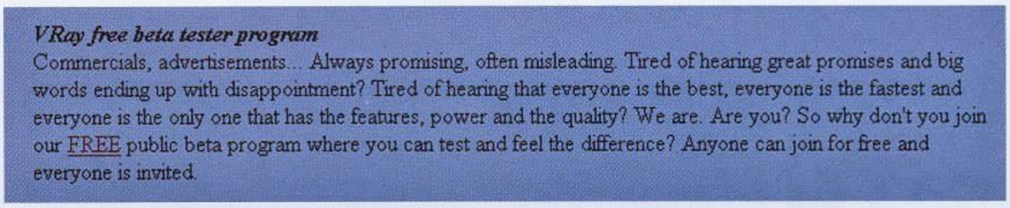
And all of these features packed into our fast raytracer engine that supports all the standard 3D Studio MAX materials environments atmospheres filters etc.

■ **Figure 2.378**
The first version of the Chaosgroup site. This page describes the features of the first version of VRay.

2001

December: the first public but **internal beta-testing** program begins. Anyone is invited to participate.

■ **Figure 1.379**
Passage taken from the site, where users are invited to take part in the beta-testing process.

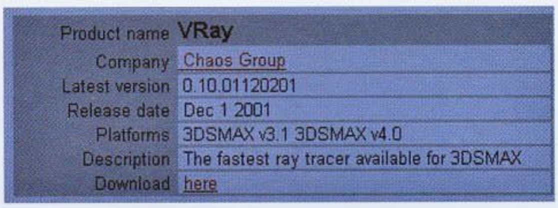

VRay free beta tester program

Commercials, advertisements... Always promising, often misleading. Tired of hearing great promises and big words ending up with disappointment? Tired of hearing that everyone is the best, everyone is the fastest and everyone is the only one that has the features, power and the quality? We are. Are you? So why don't you join our FREE public beta program where you can test and feel the difference? Anyone can join for free and everyone is invited.

2001

December: version **v. 0.10.01120201** of *VRay* is released for the beta-testing program.

■ **Figure 2.380**
Description of one of the first versions of the program.



Product name	**VRay**
Company	Chaos Group
Latest version	0.10.01120201
Release date	Dec 1 2001
Platforms	3DSMAX v3.1 3DSMAX v4.0
Description	The fastest ray tracer available for 3DSMAX
Download	here

Build History

Date	Version/download	Comments
Feb 18,2002	0.10.02021701	- minor bug fixes
Feb 04,2002	0.10.02020401	- fixed a bug in the QMC sampler that caused increased noise. - node visibility and primary/secondary visibility now work. - animated textures now work in MAX4 (really). - crash on render abort (hopefully) fixed.
Feb 02,2002	0.10.02020201	- fixed a memory leak when using DOF. - fixed the alpha channel when using DOF, and the inverted alpha in the previous build. - fixed increased noise when using secondary GI bounces. - animated textures now work.
Feb 01,2002	0.10.02020101	- timelock is extended to the end of February 2002. - VRay now uses quasi-Monte Carlo sampling for fuzzy effects (GI, DOF, glossy reflections etc) instead of the jittered grid used in the previous builds. This allows for smoother results and efficient combining of multiple effects. Direct GI is now quite a plausible alternative to the irradiance map. In fact it is the method that will produce the best results in animations. *Note:* If you use a lot of fuzzy effects, especially brute-force GI, it may be better (faster) to use the fixed-rate or two-level samplers. - some improvements in the motion blur algorithms. - the progress of irradiance map computations is displayed in the virtual frame buffer. - added some information about the irradiance map (number of samples and memory usage). - added parameters to VRayMap, VRayShadow and VRayLight to control the ray depth at which the map/shadow/light swtiches to low sampling rate. - added a parameter in the **System** roll-up to control the locking of VRay's random generator to pixel coordinates. - the diffuse illumination from VRayLights can be stored in the irradiance map for fast soft shadows.
Jan 14,2002	0.10.02011401	- fixed a bug in the stochastic motion blur sampler that caused improper rendering of transparent materials. - fixed a bug that caused artifacts with fast rotating motion blurred objects. - added a parameter for the number of geometry samples per frame for motion blur (allowing for multi-segmented motion blur); see the notes on motion blur in the documentation. - added the possibility for reusing the irradiance map for faster fly-through animations; as well as loading and saving of irradiance maps; see the notes on irradiance maps in the documentation.
Jan 10,2002	0.10.02011002	- fixed crash with translucency and irradiance map used together
Jan 10,2002	0.10.02011001	- fixed a severe memory leak in the adaptive image sampler. - motion blur is now supported. See the section on motion blur in the documentation. - fixed a problem with shading of non-uniform scaled objects.
Dec 26,2001	0.10.01122601	- timelock extended to February 1st, 2002 - fixed a bug that could cause wrong shading (red or green) of small textured objects - transparent area shadows used to cast shadows from other lights with VRay shadows, this is fixed - antialiasing filter is now On by default (it's set to Area, but you are free to choose any of the other - standard MAX AA filters). You should use filters, if you want to get the most of VRay's antialiasing system! - better atmospherics support
Dec 13,2001	0.10.01121301	- bitmapped bump maps fixed - direct lights now cast correct shadows with VRay shadows - translucency works with area shadows - refractions work with 2-sided materials turned on - light include/exclude lists work - camera clipping planes are now supported - multipliers added to GI and caustics - added a new section **Environment** that lets you override MAX's environment map and color for GI and reflections/refractions - additive transparency is now supported - pixel ratio now works - field rendering works - fixes a bug that could cause MAX to hang on four and more processor machines - optimized building of filter tables for the adaptive image sampler - the adaptive image sampler can be cancelled in the middle of a region - the Transparent parameter of VRayLight now works
Dec 05,2001	0.10.01120501	- Fixes texture coordinates on mirrored objects - alpha channel with adaptive image sampler works - Force 2-sided render option works - enhancements to VRayMap: added texmap slots for reflect/refract filter color, and a light multiplier for translucent effects
Dec 04,2001	0.10.01120401	Fixes normals (with mirrored objects), matte/shadow materials.
Dec 03,2001	0.10.01120302	Fixes on textured materials, multi-materials, crash on ray-traced shadows, secondary diffuse bounces fixed. Read the RTF file for more information.
Dec 03,2001	0.10.01120301	Major materials bug fixed. Caustics sample scene added. Secondary diffuse bounces bug fixed.
Dec 02,2001	0.10.01120201	Improved light support and bug fixes. Bump mapped refraction added. Some documentation added. Documentation is still quite poor but we promise to improve it constantly.
Dec 01,2001	0.10.0000	First version ever to appear in public.

■ **Figure 2.381**
Page from the Chaosgroup site where the first versions of the software are shown, as well as corrections made and code added.

2002

March: the **first official version** of *VRay* is presented. Pre-orders also start. By doing so it is possible to save 40% of the final price.

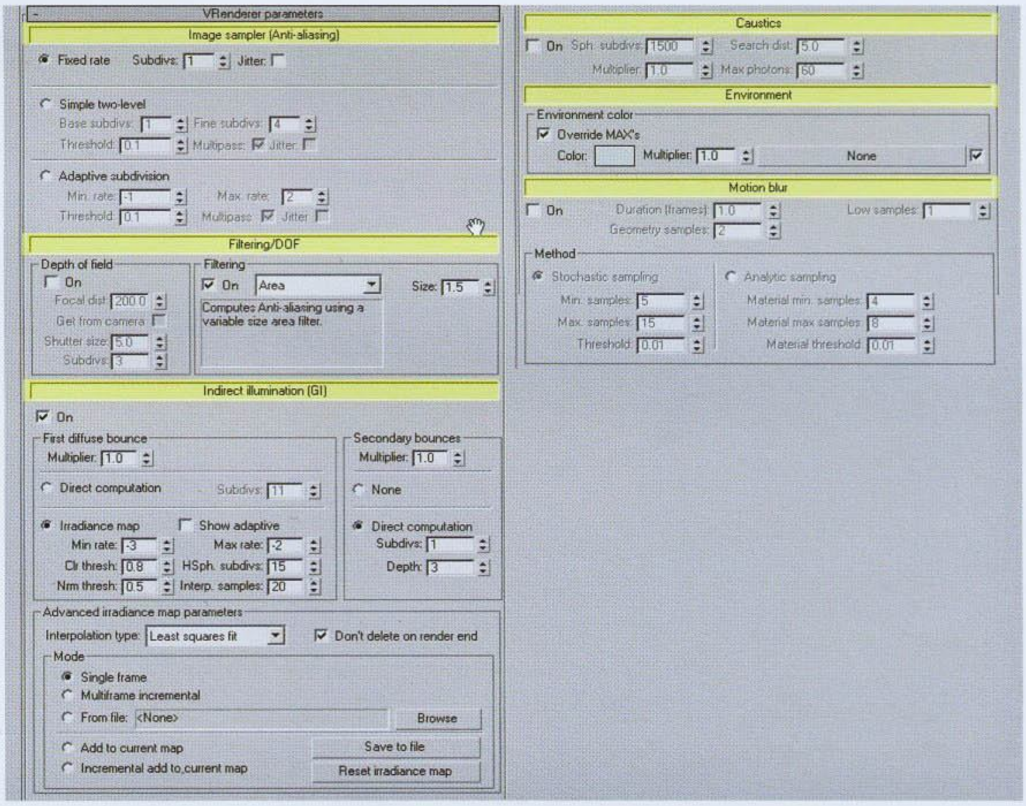
Package	Price	Description
Free	$0*	Most advanced features will be disabled. Users will still be able to render properly and enjoy the **fast** raytracer.
Basic	$299*	· True raytraced reflections and refractions · Glossy reflections and refractions · Translucency (with volume fog) for creating wax, marble, smoked glass · Area shadows (soft shadows). Includes box and sphere emitters. · Indirect Illumination (global illumination, global lighting). Different approaches include direct computation (brute force), and irradiance maps. · Motion Blur. Includes stochastic sampling approach · Depth-Of-Field camera effect. · Anti-aliasing. Icludes fixed, simple 2 level and adaptive approaches. · Caustics (Radiosity) · G-Buffer (RGBA, material/object ID, Z-buffer)
Advanced	$799*	Includes all basic features plus: · G-buffer based antialiasing · Reusable irradiance maps (save and load support). Incremental sampling for fly-through animations. · Motion blur with analytic sampling · True HDRI support. Includes *.hdr, *.rad image loader with proper texture coordinates handling for both cubic and angular maps support. Map your images directly without distortions or cropping. · Built-in area lights for physically correct illumination · Built-in material for faster materials calculations · Distributed rendering for utilizing all of your studios computers. Based only on TCP/IP enables one to connect through the internet as well. · Different camera types: fish-eye, spherical, cylindrical and cubic cameras · Network license management for using our VRay system throughout your network with less licenses.
		*all prices are in US dollars

■ **Figure 2.382**
Chart with VRay market prices and relative features.

2002

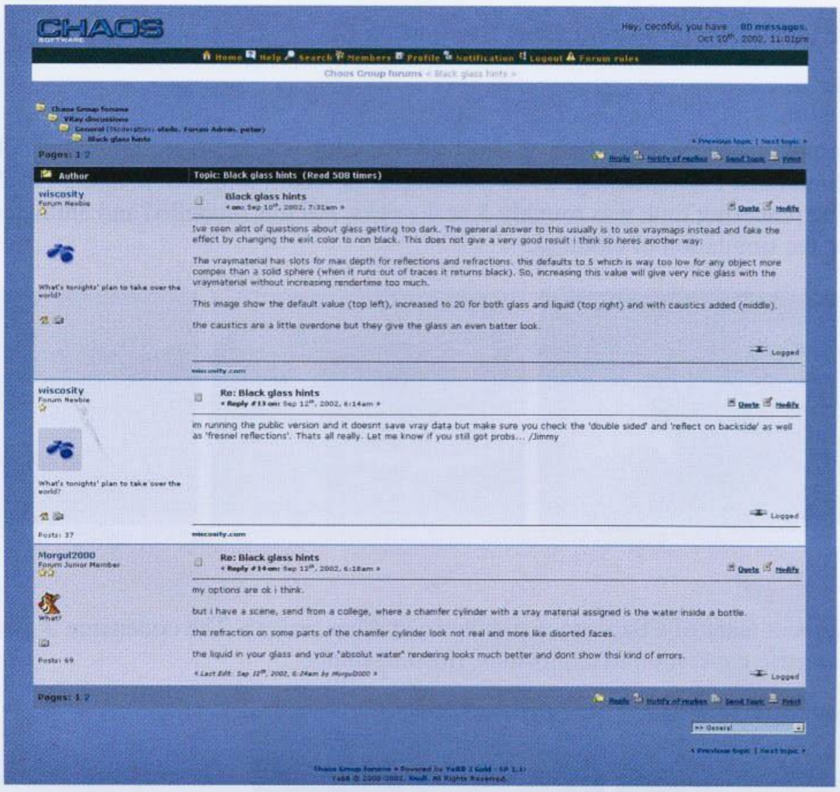
May: Chaosgroup announce the **DEMO** of *VRay* and make the download available. This version has all the features of the final release. The limitations are the presence of a watermark on the final rendering, the expiry and the fact that it is impossible to save rendering parameters in the Max file.

■ **Figure 2.383**
Images of the Renderer panel of one of the first versions of VRay.



2002

June: the **VRay forum** is born. The aim is to give thorough and real support for Chaosgroup products. Anyone can register and take part in any thread discussion. The forum is based on the popular YaBB system.



■ **Figure 2.384**
The first version of the VRay forum.

2002

June: Chaosgroup announce that they will give a 25% **discount** on the catalogue price of their top program. The Basic Version costs 224$ instead of $299 and the Advanced Version costs $559 against $799. The offer remains valid only until the end of the SIGGRAPH event.

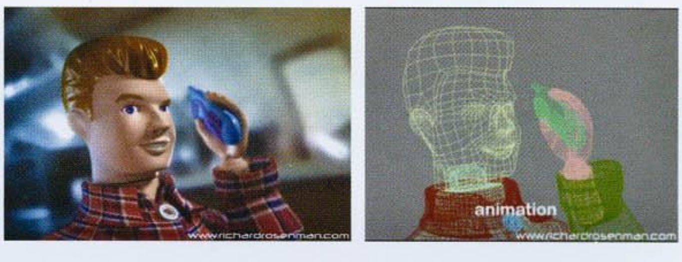
2002

October: after a long wait, the first real **FREE** version of *VRay* is released, freely downloadable from the site. It includes:

- Calculation of reflections and refractions.
- Calculation of indirect illumination, commonly known as Global Illumination (GI).
- Depth of Field.
- Quasi Monte Carlo sampling (QMC).
- Varied antialiasing filters.

2002

August: **Richard Rosenman** creates his first 60" animation called **Covena** using **VRay 0.10.01121301**. Modelled and rendered with 3ds Max 4.2, it requires a good 30 minutes per frame, using a Dual P3 at 1 GHz. It is the beginning of a long brothership which leads to many works, amongst which the short film, Plumber.



■ **Figure 2.385**
Images from the animation of Covena.
@2002 Richard Rosenman.

2002

December: Chaosgroup announce a 20% discount for all of the Christmas period.

2003

January: **Digital Dimension** uses *VRay* for its film **Final Destination II**.

2003

February: **Richard Rosenman** produces **Plumber**, a short film completely rendered in **VRay v1.09**. The whole development took Richard 6 months. For rendering, 20 to 30 computers, dual P3 at 1 GHz and P4 at 2 Ghz were used, each one with 1 GB Ram. The rendering time was roughly 3 or 4 weeks. All this modelled and animated in 3ds Max 4.2 with the help of Simcloth for the simulation of clothing.

■ **Figure 2.386**
- Advertising playbill of the short film.
- Image taken from the animation.
- Texture of the protagonist's face.
@ 2003 Richard Rosenman.

2003

March: the software development team of Chaosgroup announce another project. The codename is **Aura**, a plug-in for fluid simulation, for the volumetric generation of explosions, fire and smoke.

■ **Figure 2.387**
Some of the scenes it is possible to make with Aura.
@2003 Chaosgroup.

2003

May: the third freeware plug-in version of **Simcloth**, developed by Vladimir Koylazov (Vlado) is released. It is a program for the simulation of clothing and objects with a soft or hard consistency. Simcloth supports both collision between objects and between cloth and cloth. Simcloth v3.0 adds considerable improvements, such as a faster, more stable and more accurate simulation. It is an open-source plug-in and thus it is free to be modified.

■ **Figure 2.388**
Some of the possible scenes which can be made with Simcloth.

2003

May: **Rezn8** creates a 30" advertisement for **Alienware** using *VRay* and 3ds Max.

■ **Figure 2.389**
Frames captured directly from the original animation.

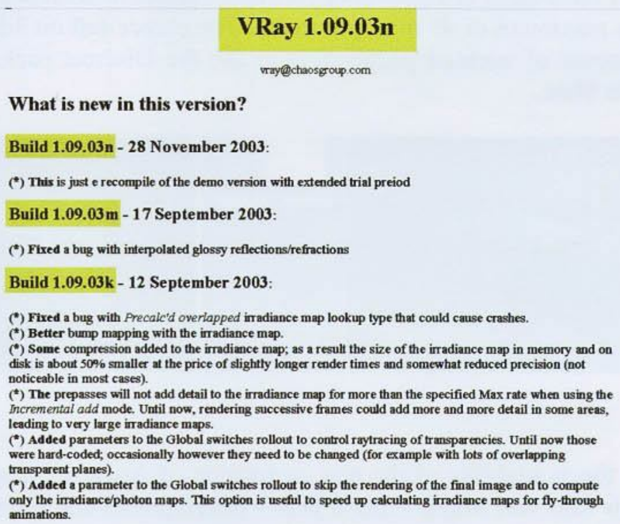
2003

May: by this time, *VRay* already possesses a vast array of features:

- Physically accurate reflections and refractions.
- Blurred reflections and refractions.
- Properties of translucence of materials, called Subsurface Scattering (SSS), for the creation of wax, marble etc.
- Generating soft shadows, called Area Shadows, with the chance of using two types of emitters, sphere and box type
- Calculation of indirect illumination via three types of calculations: Irradiance Map, which makes it possible to approximate the computation of GI, making it faster but less precise; Brute Force, or direct method, very precise but also very slow, and Photon Mapping.
- Physically correct Motion Blur for obtaining realistic moving objects.
- Depth of Field for obtaining focus of foreground subjects or objects in the background out-of focus.
- Anti-aliasing with two available systems for calculations, one based on QMC and the other on an adaptive process.
- Caustics.
- G-Buffer (RGBA, material/object ID, Z-buffer).
- Possibility of re-using the Irradiance Map for multiple scenes and Fly-through animations.
- HDRI image support.
- Area lights for a correct illumination.
- Availability of proprietary shaders, as well as the standard 3ds Max ones, for faster and more accurate rendering.
- Distributed Rendering.
- Displacement Map, for creating micro-details in the rendering phase.
- New cameras: Fish-Eye, Spherical, Cylindrical and cubic.

2003

September: version **v1.09.03n** of *VRay* is released. For almost a year it is the only available version, before the introduction of version 1.45.70 in July 2004, which will bring about significant updates.



VRay 1.09.03n

vray@chaosgroup.com

What is new in this version?

Build 1.09.03n - 28 November 2003:

(*) **This** is just e recompile of the demo version with extended trial preiod

Build 1.09.03m - 17 September 2003:

(*) **Fixed** a bug with interpolated glossy reflections/refractions

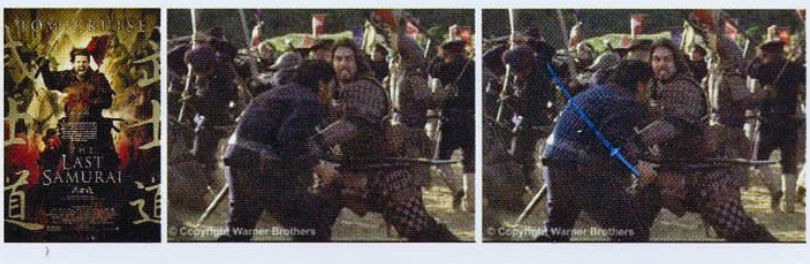
Build 1.09.03k - 12 September 2003:

(*) **Fixed** a bug with *Precalc'd overlapped* irradiance map lookup type that could cause crashes.
(*) **Better** bump mapping with the irradiance map.
(*) **Some** compression added to the irradiance map; as a result the size of the irradiance map in memory and on disk is about 50% smaller at the price of slightly longer render times and somewhat reduced precision (not noticeable in most cases).
(*) **The** prepasses will not add detail to the irradiance map for more than the specified Max rate when using the *Incremental add* mode. Until now, rendering successive frames could add more and more detail in some areas, leading to very large irradiance maps.
(*) **Added** parameters to the Global switches rollout to control raytracing of transparencies. Until now those were hard-coded; occasionally however they need to be changed (for example with lots of overlapping transparent planes).
(*) **Added** a parameter to the Global switches rollout to skip the rendering of the final image and to compute only the irradiance/photon maps. This option is useful to speed up calculating irradiance maps for fly-through animations.

■ **Figure 2.390**
Description of the builds previous to v1.09.03.

2004

March: **Digital Dimension** uses *VRay* for 50 shots in the film **The Last Samurai**. The choice to use Chaosgroup's product was taken because of the great quality of the Motion Blur and out-of-focus reflections.



■ **Figure 2.391**
The playbill and images taken from the film The Last Samurai. @2004 Digital Dimension.

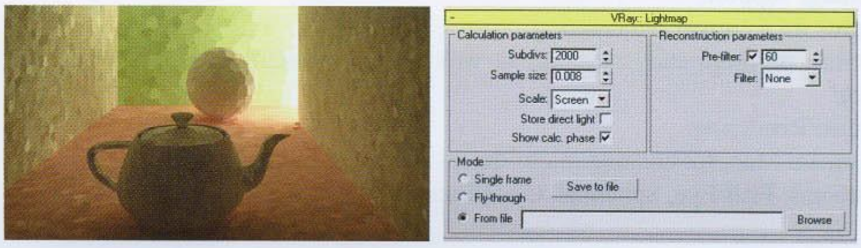
2004

March: **Ryuichi Yagi**, art director of Shirogumi, Tokyo, decides to use *VRay* in the cinematic opening of the game **Onimusha 3**. After trying different engines, Yagi reports that the decision to use *VRay* was due to the quality of its Motion Blur, rendering speed and production of photorealistic highlights. Instead for the creation of hair the program Shag:hair was used.

■ **Figure 2.392**
Cover of the DVD of the game, and images captured from the introductory video. @2005 Shirogumi.

2004

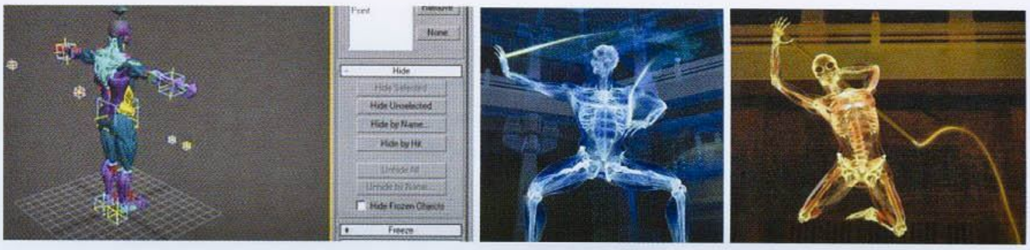
March: Chaosgroup announce the use of a new GI calculation system through the **Lightmap**, now called Light Cache, which is presented in v1.5. It is a method similar to Photon Maps, but without the countless number of limitations that this had.



■ **Figure 2.393**
Example of a Lightmap calculation and the control panel. @2004 Chaosgroup.

2004

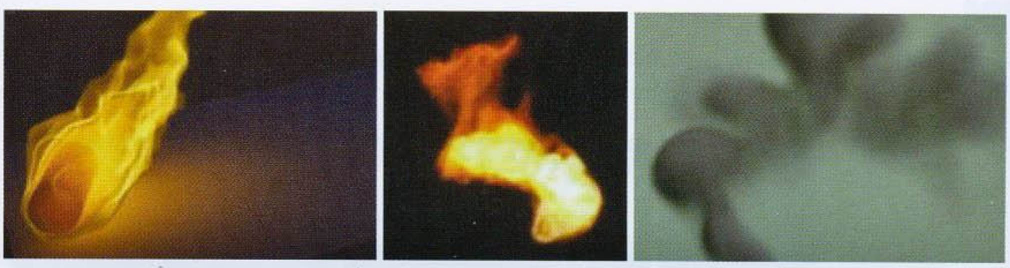
March: **Animal Logic** uses *VRay* in a documentary for the **Discovery Channel**. The main problem was the reproduction of transparency in the bodies of fighting opponents. Thanks to its renderfarm, the rendering time varied from a minimum of 5 min. to a maximum of 45 min. per frame. The choice fell on 3ds Max 5 mainly for two reasons: The first was the enormous amount of talented people able to use the Discreet package well; the other was because *VRay* was available only for 3ds Max.



■ **Figure 2.394**
- 3ds Max screenshot of one of the two characters.
- Frames of some parts of the video.
@2005 Animal Logic.

2004

March: Chaosgroup announce the availability of the new generation of **Aura**, its plug-in for the simulation of fluids, fire and smoke available for 3ds Max and Maya. After a year's development and rewriting the code, it is available for **internal beta-testing**.



■ **Figure 2.395**
Examples of renders generated with Aura.
© 2004 Chaosgroup.

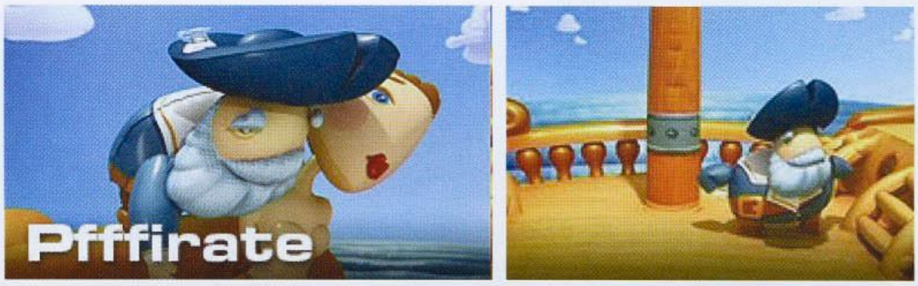
2004

June: as a personal project **Scanline**, a German society based in Munchen, developed some tools for simulating and rendering fluids. The current project is based on **implicit surfaces** and volumetric effects. This was possible thanks to an intense use of **VRay's SDK**, made available by Chaosgroup, and thanks to the incredible support given by the latter.
For the generation of an image of the surface of a sea, initially the idea was to use polygons, but in order to obtain a realistic effect, millions of them would have been necessary. At this point, the idea cropped up to use Implicit Surface to represent the ocean. With this method, surfaces are not represented through a grid of vertexes, but by mathematical formulae. This information is then used by a raytracer to reconstruct the surfaces with the same properties as possessed by polygons (normal, UV coordinates) and similar behaviour (Global Illumination, caustics, reflections). This way the amount of memory required is significantly less. Scanline has created a dynamic ocean generator, called **Oceanline**. It contains all the characteristics of sea surfaces, including waves and crests.
In this same way, **Fogline** was developed, a back lighting effect generator which works in a physically true fashion, and is not a simple Glow in 2D post-production. These are the first two plug-ins for *VRay* and, according to Scanline, will not be the last.

■ Figure 2.396
- Scene elaborated with Oceanline.
- Scene elaborated with Fogline.
- Scene created via Implicit Surface.
- Original scene without the application of Implicit Surface.
@2004 Scanline.

2004

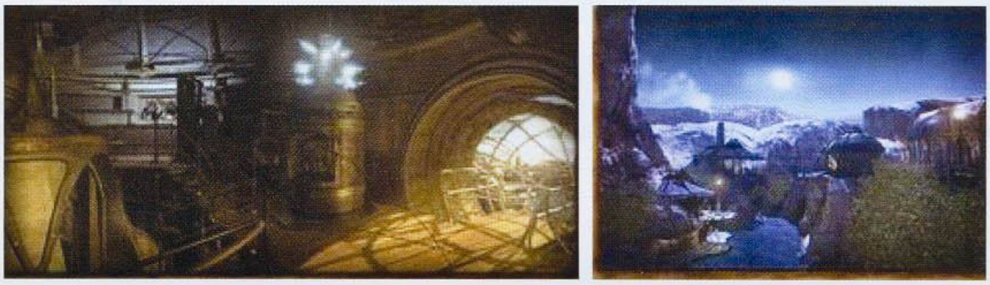
June: **Pfffirate** wins the Best Student 3D Animation prize at the 3D Awards. It is modelled and animated completely through 3ds Max and rendered with *VRay*.



■ Figure 2.397
Images taken from the animation Pfffirate.
@2004 Xavier André.

2004

June: **Myst IV REVELATION** is completely created and rendered with *VRay*. The Scanline was used for the preview and approvation on behalf of the team and *VRay* was used for thousands of frames in the game. Just to give an idea, the average size for each scene was 300 MB, made of 3,000 objects, 2,000,000 polygons, 600 textures, 50,000 frames which make use of many other plug-ins compatible with *VRay* and a great number of light sources, using both ShadowMap and VRayShadow. As far as the follow-up is concerned, no information is available but from what has been said, *VRay* will be used once more.



■ Figure 2.398
Some of the wonderful locations in Myst IV.
@2004 Cyan Studios.

2004

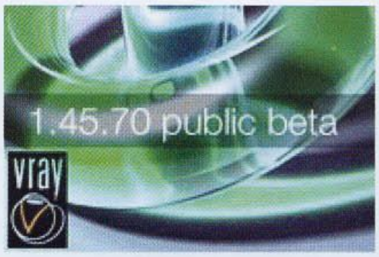
June: at SIGGRAPH a first version of **VRay v1.5** is presented, and also the beta version of **VRay Standalone** and **VRay for Maya**.

2004

July: The much awaited **v1.45.70** version of *VRay* is released. The main additions are:

- Faster Displacement.
- VRayFur for body hair creation.
- VRayToon for hidden-lines rendering.
- VRayProxy, an instance system generation for objects with a large number of polygons.
- Improved Motion Blur.
- New license management system.
- VRay Frame buffer, a proprietary frame buffer.
- Channel rendering, for rendering separate channels. Now shadows, reflections, refractions and so on can be rendered on different levels, as when using Rendering Elements in 3ds Max.
- Improvements in the shaders, such as: separate IOR between reflections and refractions, "unnatural" Highlights, the presence of an opacity map which was not present before, a new shader, VRayLight, for the generation of self-illuminating objects.
- New system for anti-aliasing sampling.
- Lightmaps, a new GI calculation system.
- Management of saturation and contrast of GI.
- Improved DOF.

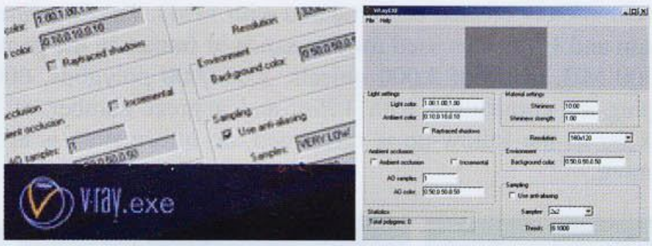
■ **Figure 2.399**
VRay v1.45.70 public beta.
@2005 Chaosgroup.



2004

July: **VRay.exe**. It is the first project made public by Chaosgroup of the Standalone version of *VRay*. It is still rudimental and does not permit any type of professional work to be carried out on it. It basically only allows one to import primitives, global management of highlights and AA settings. The only GI calculation that can be done is by using sources such as Environment, which can be color modified, and a preset omni which can be color modified but not the position.

■ **Figure 2.400**
VRay.exe, the first public version of the Standalone version of VRay, although in practice almost unusable.



2004

September: the beginning of the public beta-testing of **Aura** is announced. Finally, after months of beta-testing "behind closed doors", Aura becomes **public** for 3ds Max users.

2005

May: the last available version before v1.5 is released, at least as far as public users are concerned. It is the famous **v1.47.03**.

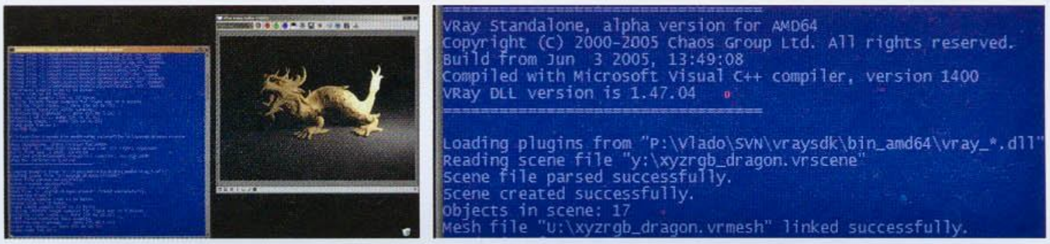
2005

June: the new versions of **VRay 1.5** are presented at the **SIGGRAPH** with new features (Sun/Sky system, physical camera, extended shaders), as well as **VRay for Maya**, **VRay Standalone**, **VRay for Truespace 7**, **Aura**, and for the first time, **VRay for Rhino**, thanks to the tight collaboration with ASGvis.
The main novelties are in **VRay 1.5**. This time lots of information and screenshots of the much awaited version are released:

- Introduction of Motion Blur in the particle system of 3ds Max, Particle flow.
- Introduction of Image-based lighting (IBL).
- Improvement of the QMC system.
- Improvement of Photon Mapping.
- Introduction of the new "Sun" and "Sky" system for correct lighting of outdoor and indoor spaces.
- Introduction of new shaders, called Dirt shaders, that is, Ambient Occlusion and a new system, much faster than the previous one, for the calculation of SSS.
- New cameras based on physical values, such as f-stop, focus distance, etc…

2005

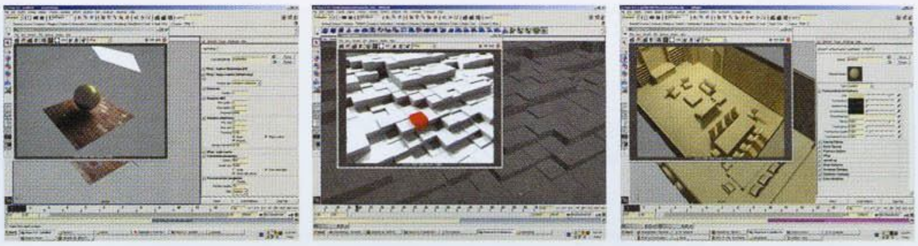
June: for the first time screenshots are shown of the 64 bit version of **VRay Standalone**.



■ **Figure 2.401**
Screenshots of the 64 bit version of VRay.
@2005 Chaosgroup.

2005

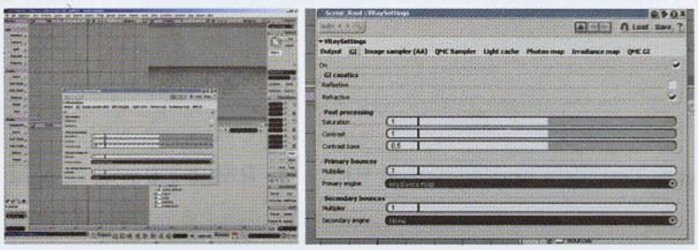
June: for the first time screenshots are shown of the **VRay for Maya**.



■ **Figure 2.402**
Screenshots of VRay working in Maya.
@2005 Chaosgroup.

2005

June: for the first time screenshots are shown of the **VRay for XSI**.



■ **Figure 2.403**
Very first screenshots of VRay working on XSI.
@2005 Chaosgroup.

2005

September: once again, **Richard Rosenman** astounds all with his amazingly high quality advertisements. The design is inspired by the film "The Incredibles", the two advertisements show the quality which can be reached with the last version of **VRay 1.47.03**. The development required 2 months' work in a team formed by 6 people. The software used is 3ds Max 6, AutoDesk's Combustion 4.0 and Edited Adrenalin by Avid.

■ **Figure 2.404**
Wonderful animation by the team, lead by Richard Rosenman.
@2005 Richard Rosenman.

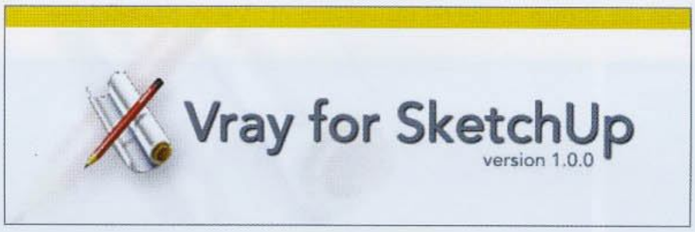
2005

November: the **beta-testing phase for Maya** becomes reality. Like all testing phases, though, access is closed. Anyone can participate by sending a mail to Chaosgroup, giving a reason and sending some of their own works. A reply e-mail is then sent, if in accordance, for entry in the Beta-testing group.

2005

November: **ASGvis**, a company in Baltimore, is a Computer Graphics firm which develops software, plug-ins and exporters for 3d applications. It announces that **VRay for SketchUp v1.0.0** is in its last stage of development.

■ **Figure 2.405**
Advertisement logo of VRay for Sketchup.



2005

December: **ASGvis** announces the beginning of the closed beta-testing phase for **VRay for Rhino 1.0.0**. This is the last phase before marketing the development of this plug-in.

■ **Figure 2.406**
- Publicitary logo of VRay for Rhino.
- Image made with this version.

2006

April: an Italian, **Renato Taraballa**, announces the beginning of his project. Namely the porting of **VRay for Cinema 4D**.

2006

January: **ASGvis** announces that for a limited period **VRay for Rhino 1.0** will be sold for only $499.

2006

July: **Chaosgroup** announces the presentation of **VRay 1.5** at the Boston SIGGRAPH. The wait is over! The official pre-release of VRay 1.5 works only with a dongle, a USB hardware key containing the license of the program.

2006

July: **ASGVIS** starts the official sales of **VRay for Rhino 1.0**.

2006

August: **Chaosgroup** makes the **DEMO v1.5 RC2** of *VRay* available.

2006

October: **Chaosgroup** makes the **v1.5 RC3** of *VRay* available, both 32 and 64 bit.

2006

October: **Chaosgroup** makes the **v0.84.22 VRay4Maya** version of *VRay* available.

2007

January: **Chaosgroup** releases the **v1.5 RC4** of *VRay*.

2007

May: **ASGVIS** releases the new version of **VRay for Sketchup 5 and 6**.

2007

June: **Chaosgroup** makes the **v0.85.10 VRay4Maya** version of *VRay* available.

2007

June: **Dieter Morgenroth** releases version **0.3 beta** of his plug-in **VRay ISO** for *VRay*. It is a system for generating surfaces and geometries in a parametric way, based on textures.

2007

August: **Chaosgroup** will be present at this year's **SIGGRAPH** with big news. Versions will be presented for Maya, XSI, Standalone and Rhino, as well as version **1.5 FINAL**.

2007

September: version **1.0** of **VRay** is released for **Cinema4D**.

2007

September: **Chaosgroup** release version **1.5 FINAL** of *VRay*. After so many years of waiting, finally the official version is available to the large public.

2008

March: **Chaosgroup** have posted a few new videos showing their RT rendering (**Real Time**) used as an active shade renderer in 3ds max.

2008

April: **Service Pack 2** for **VRay 1.50** is now available for download.

CHAPTER 3: RENDERER - PART 1

The Render tool can be activated by clicking on the F10 button, or alternatively through the path *Rendering --> Render -Renderer Tab*. It is the panel containing most of V-Ray's parameters for the management of illumination, cameras, antialiasing, etc.. Because of the large amount of parameters and rollouts contained in it, this subject will be examined in two different chapters.

The V-Ray Frame Buffer (VFB) represents the window through which V-Ray shows the user the final output, that is to say the render. It is thought specifically for V-Ray, but can also be activated in the place of the 3ds Max Frame Buffer. It has many features not present in 3ds Max one, parameters which can be useful in several situations. In the following paragraphs we will see them. The VFB is represented by two separated tools. The first one is the rollout V-Ray: Frame buffer, which enables its management and activation.

Figure 3.1

The second one can be considered as the real VFB, represented by a window appearing at the moment of the rendering starting.

Figure 3.2

Inside VFB other sections are present, such as the Color corrections, which can be activated by clicking on the first icon located down on the left of V-Ray's VFB. This section allows one to adjust colors as if one were using a photo-retouching program.

Figure 3.3

Those who are familiar with 2D will have surely noticed the similarities in this panel with some tools also present in various bi-dimensional image editing programs.

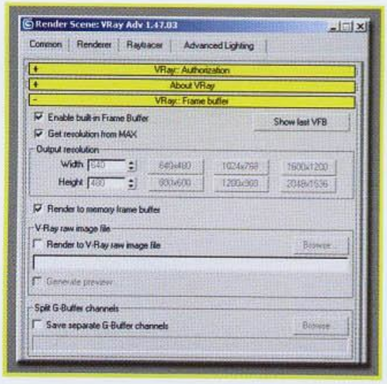
VRAY: RENDERER - PART 1

The Render tool can be activated by clicking on the F10 button, or alternatively through the path ***Rendering -> Render ->Renderer Tab***. It is the panel containing most of *VRay*'s parameters for the management of illumination, cameras, antialiasing, etc... Because of the large amount of parameters and rollouts contained in it, this subject will be examined in two different chapters.

VRay: Frame buffer

INTRODUCTION

The ***VRay Frame buffer*** (***VFB***) represents the window through which *VRay* shows the user the final output, that is to say the render. It is thought specifically for *VRay*, but one can use it instead of the common 3ds Max Frame Buffer. It has many features not present in 3ds Max one, parameters which can be useful in several occasions. In the following examples we will see them. The ***VFB*** is represented by two separated tools. The first one is the rollout ***VRay: Frame buffer***, which enables its management and activation.



■ **Figure 3.1**
The ***VFB*** is a valid alternative to 3ds Max one.

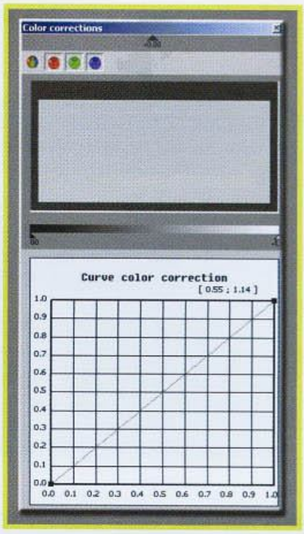
The second one can be considered as the real ***VFB***, represented by a window appearing at the moment of the rendering start-up.

■ **Figure 3.2**
The ***VFB*** also allows a management separated into ***layers***.

Inside ***VFB*** other sections are present, such as the ***Color corrections***, which can be activated by clicking on the first icon located down on the left of *VRay*'s ***VFB***. This section allows one to adjust colors as if one were using a photo-retouching program.

■ **Figure 3.3**
In ***VFB***'s lower part many parameters are present.

Those who are familiar with 2D will have surely noticed the similarities in this panel with some tools also present in various bi-dimensional image editing programs.

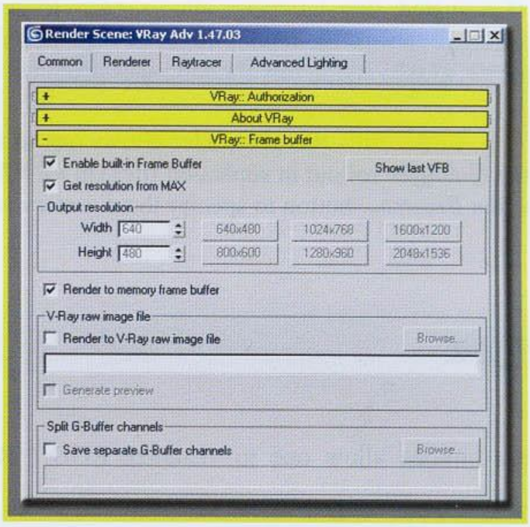


■ **Figure 3.4**
The ***Color corrections*** panel is divided in 3 areas: the first line, containing the little overturned triangle, allows one to modify the exposure, the central panel to correct levels and the grid chart down the image correction by means of curves. All changes carried out with ***Color corrections*** tools will be present in the saved render.

PARAMETERS

In the following pages each single parameter included in the ***VFB*** will be described and analyzed, starting from the rollout ***VRay: Frame buffer***.

Frame buffer



■ **Figure 3.5**
VRay: Frame buffer is one of the 15 rollouts in the Render tool of 3ds Max.

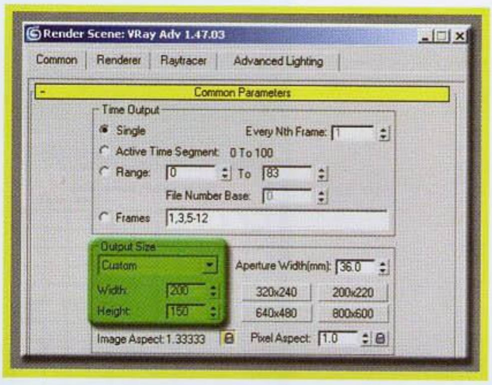
Enable built-in Frame buffer - this parameter activates *VRay*'s ***Frame buffer***.

For technical reasons, the 3ds Max Frame buffer will continue to be present, but will not give back any image. It is advisable to manually disable the 3ds Frame Buffer with the appropriate option present in the section Render Output of 3ds Max, inside the Common tab. This operation allows to regain memory (RAM) useful for rendering in *VRay*, because it will not be used anymore for the image inside the Frame Buffer in 3ds Max. With version v1.5 RC3 this last operation is not necessary anymore: in fact activating *VRay*'s ***Frame buffer***, 3ds Max Frame Buffer will be automatically disabled.

■ **Figure 3.6**
The 3ds Max default setting provides for the activation of the Frame Buffer.

Get resolution from MAX - thanks to this parameter, the output resolution is the one set inside the 3ds Max Common panel.

■ Figure 3.7
3ds Max Output Size panel has a few more functions compared to VRay's one.



Output resolution - this section, very similar to the one in 3ds Max, enables one to set the size of render output, regardless of the settings in the Common Parameters rollout.

Render to memory - this parameter activates ***VFB***'s visualization, a window allowing one to look at the image while the render is computed. When high-resolution images are being computed it is best to disable it. In this way part of the RAM, otherwise no longer available, will be saved, because of being used by the ***VFB*** itself. In case of deactivation one might use the parameter ***Render to V-Ray row image file***.

Render to V-Ray image file - when activated, *VRay* creates a ***.vrimage*** file, corresponding to the image's final output. Not much RAM is used for its creation, leaving it available for other purposes. To see what *VRay* is creating it is advisable to activate the option ***Generate Preview***.

Generate Preview - if activated, this parameter allows *VRay* to create a little preview of what it is rendering. In case the ***VFB*** is deactivated, a preview of the render might be observed as well, allowing the user to analyze it, and stop it in case of mistakes. In the v1.47.03 version this function was disabled.

Save separate G-Buffer channel - this parameter allows one to save the channels of a render separately, layers which can be selected in the ***G-Buffer*** rollout in separated files (in ***v1.5 RC3*** the ***G-buffer*** has been deleted and replaced by ***Render Elements***). Use the ***Browse...*** button to specify the directory where you want to save it.

Frame buffer toolbar

 - these buttons allow one to visualize RGB (Red, Green, Blue), alpha and monochromatic channels.

 - parameter which allows one to save the render as an image file (.jpg, .bmp, .tiff, .hdr, etc….). It is possible to save the image during the creation of the render.

 - parameter which deletes the residual image within the ***VFB***. It is useful when the previous render causes confusion.

 - parameter which allows one to copy the image of the ***VFB*** inside the 3ds Max Frame Buffer. It is possible to carry out this operation during the rendering phase too.

 - when activated, this parameter allows one to render the part of the image manually selected with the mouse. Moving the mouse on the ***VFB*** one is able to compute the desired part during the rendering computing.

 - parameter which activates and visualizes the ***Color corrections*** panel of *VRay*.

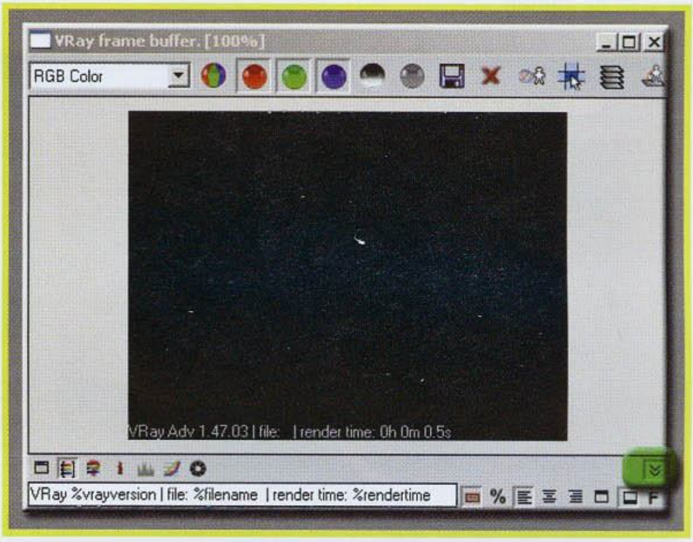
- this tool activates a panel where all the information about pixels composing the render is contained. It is possible to make this tool appear also by clicking the right mouse button (**RMB** = **R**ight **M**ouse **B**utton) inside the ***VFB***. In case the button is not activated, by clicking the RMB inside the ***VFB***, the ***Pixel information*** panel can be toggled to appear by clicking the mouse, providing the information required.

- if active, this button allows one to visualize the changes carried out on the levels and fulfilled with the tools of the panel ***Color corrections***.

- if active, this button allows one to visualize the changes carried out on the curves and fulfilled with the tools of the panel ***Color corrections***.

- if active, this button allows one to visualize the changes carried out on the exposure and fulfilled with the tools of the panel ***Color corrections***.

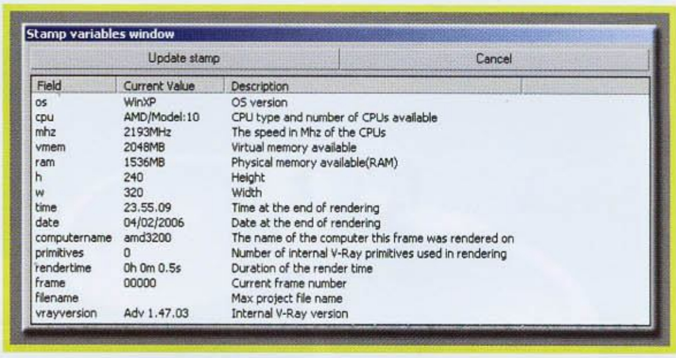
- if active, this button makes a new toolbar appear in the lower part of the ***VFB***. These tools allow one to change the textual line containing information on the rendering in progress.



■ **Figure 3.8**
With the interactive ***Frame stamp*** it is possible to modify the text row very quickly.

- button which activates or deactivates the ***Frame stamp,*** which is the lower line in the render containing information on the render in progress.

- parameter which activates a new window. Inside it all the variables which can be visualized in the ***Frame stamp*** are present.



Field	Current Value	Description
os	WinXP	OS version
cpu	AMD/Model:10	CPU type and number of CPUs available
mhz	2193MHz	The speed in Mhz of the CPUs
vmem	2048MB	Virtual memory available
ram	1536MB	Physical memory available(RAM)
h	240	Height
w	320	Width
time	23.55.09	Time at the end of rendering
date	04/02/2006	Date at the end of rendering
computername	amd3200	The name of the computer this frame was rendered on
primitives	0	Number of internal V-Ray primitives used in rendering
rendertime	0h 0m 0.5s	Duration of the render time
frame	00000	Current frame number
filename		Max project file name
vrayversion	Adv 1.47.03	Internal V-Ray version

■ **Figure 3.9**
Numerous variables can be visualized inside the ***Frame stamp***. Keeping ***SHIFT*** pressed and clicking on the variables it is possible to add or delete them in order with user's choice. The dimension of the text line will fit consequently.

- parameters which allow the alignment of the ***Frame stamp*** text on the left, the right or in the middle.

- parameters which allow one to put the text line high or low within the render.

- parameter which allows one to visualize the window " character types" of Windows.

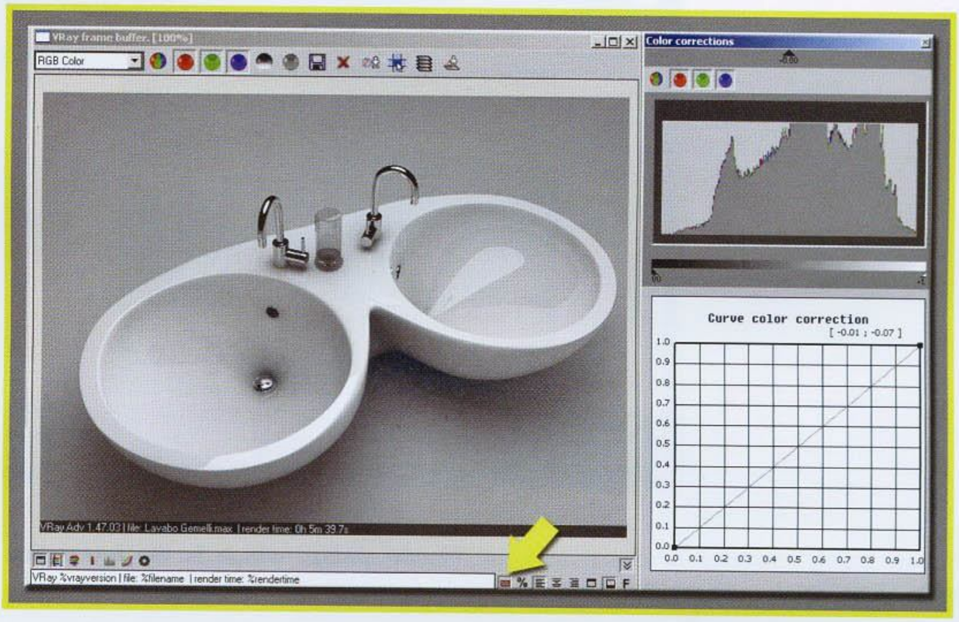
Frame buffer shortcut

Here below a list of keyboard shortcuts to be used inside the ***VFB*** will be shown.

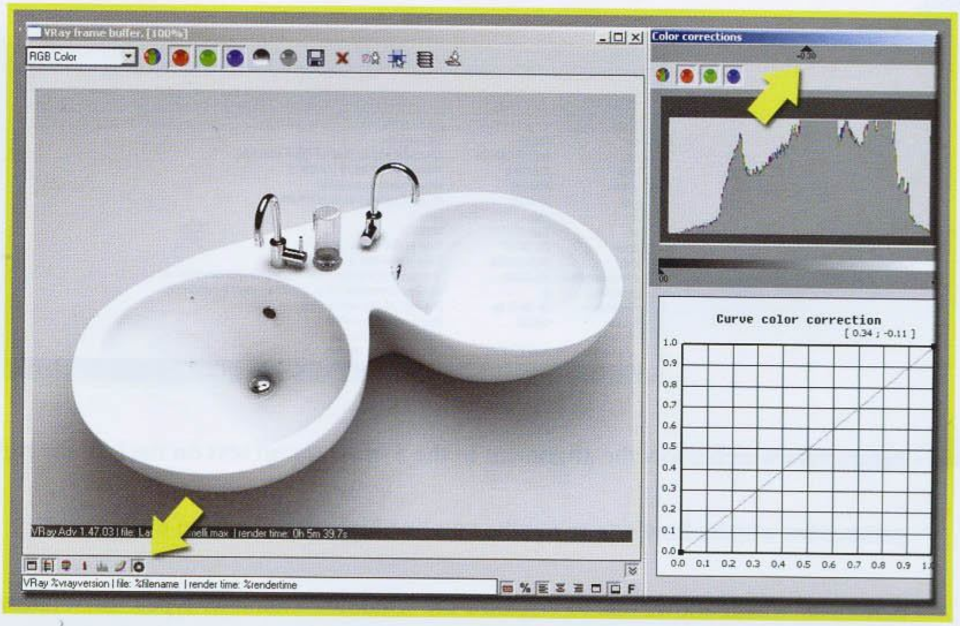
Mouse	Description
CTRL + LMB CTRL + RMB	Positive Zoom. Negative Zoom.
Central wheel rotation	Positive Zoom / Negative Zoom.
Double click LMB	100% Zoom.
RMB	It shows the dialogue panel which gives information concerning the selected pixel.
MMB	Pan.

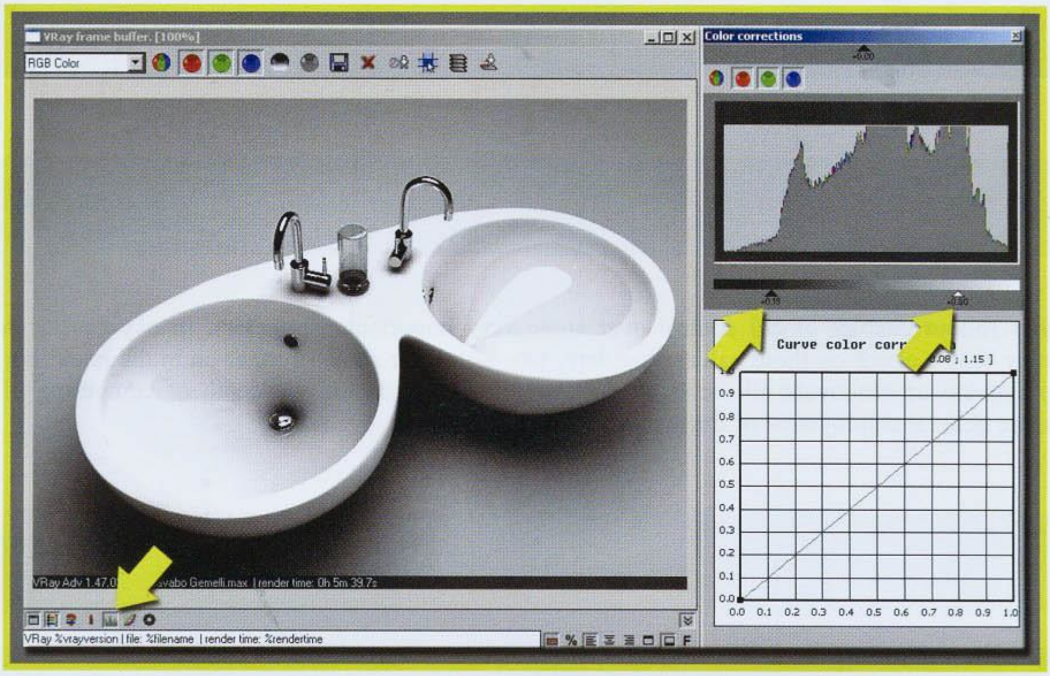
Some examples of ***VFB*** use will be shown.

■ **Figure 3.10**
In this scene all the default values have been kept the same, except for the activation of the ***Frame stamp***. Thus it is possible to introduce and obtain information concerning the render.



■ **Figure 3.11**
In this scene there are only default values, except for the parameters setting the exposure. As pointed out by the arrows, to make the correction visible one has to activate the button on the lower-left inside the ***VFB***. In this case, as a consequence of the changes carried out, the image appears brighter.





■ **Figure 3.12**
This image has been adjusted working on levels. Now the image has more contrast.

■ **Figure 3.13**
Curve corrections through the *VFB*.

VRay: Global switches

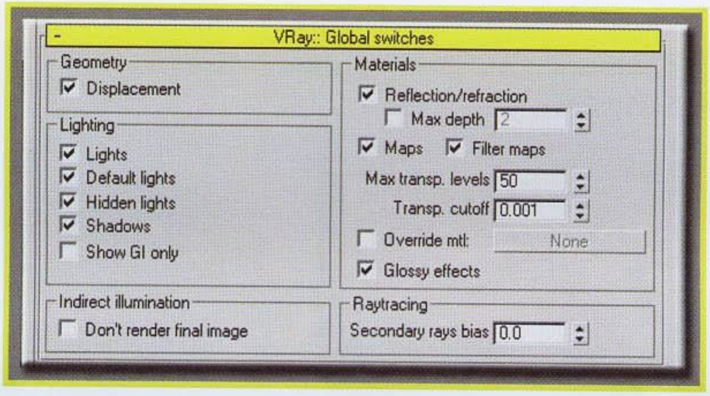
INTRODUZION

The *VRay: Global switches* rollout enables global control over some very important parameters. In any case it is possible to find again, edit or keep a check on these values singularly too, as will be shown further on in the book. For example, it is possible to activate or deactivate all the lights just by a click, or decide the depth of computing of reflections and refractions, the general employment of the textures, etc...

PARAMETERS

Here below are all the parameters composing the *VRay: Global switches* will be analyzed.

■ **Figure 3.14**
The *VRay: Global switches* rollout is a global system to handle some of the most important rendering parameters.



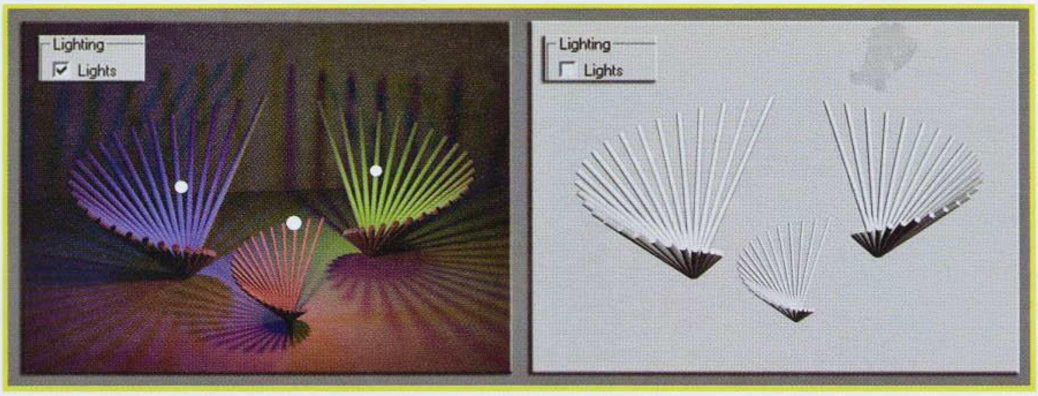
Geometry

Displacement - activates or deactivates *VRay*'s *Displacement*. This parameter has no effect on 3ds max *Displacement*, which will remain active. To disable the one in 3ds Max it is necessary to use the corresponding option present in the *Common* tab of the panel *Render*.

■ **Figure 3.15**
The *Displacement* is an effect allowing one to add details and modify geometries only in the rendering phase.

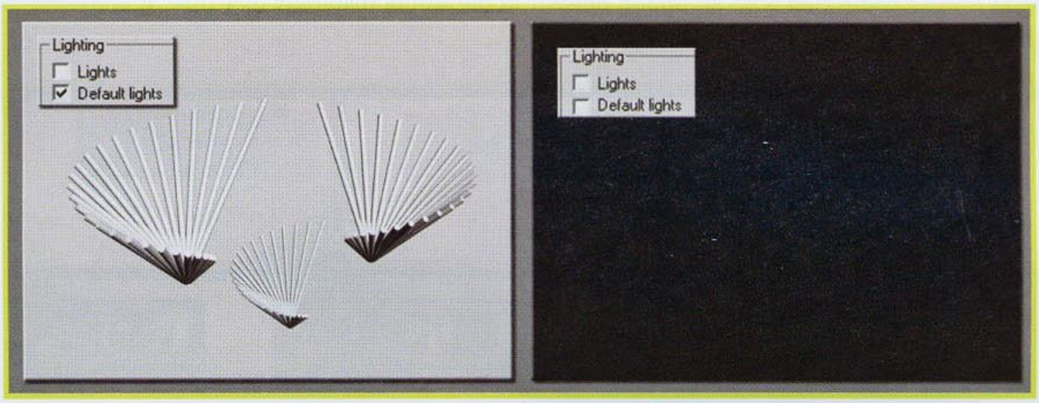
Lighting

Lights - it activates or deactivates all the lights present in the scene. One must remember that in case this parameter is deactivated, the *On* option of each single light within the Modify panel is kept active. Deactivating the *Lights* parameter, the complete deactivation of any light in the render will be obtained. 3ds Max default light remains active and *VRay* uses it for the scene's illumination each time a render is started.

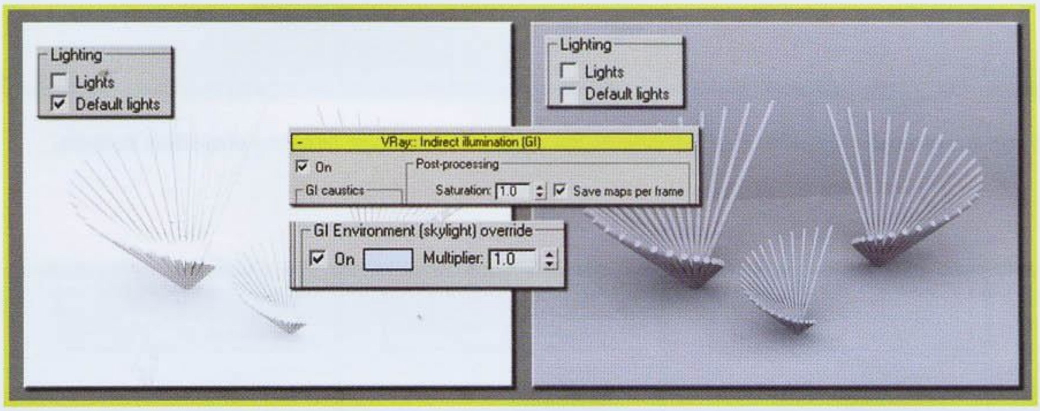


■ **Figure 3.16**
Three colored ***VRayLights*** illuminating simple medium-grey objects. The deactivation of the option ***Lights*** switches off all the lights. No GI has been activated in the scene. The light illuminating the meshes in the image on the right is the 3ds Max default lighting, a global light not containing parameters nor being editable. It is used just for "seeing" objects in viewport in order to avoid them from being completely dark.

Default lights - parameter which activates or disables *VRay*'s use of 3ds Max default lights. It is compulsory to deactivate this option when the scene is lighted just by the ***skylight***, in order to obtain a correct render. In case one forgets to deactivate it, *VRay* uses, in addition to ***skylight***, the 3ds Max default light as well. In this way the render is wrong. Another solution, always valid when ***skylight*** is used as the only luminous source, is the creation of a generic light of 3ds Max or *VRay*. It will then be a user's task to deactivate it. This last operation makes 3ds Max switch off the default light automatically, as it has become useless because of the presence of a new luminous source. The employment of the parameter ***Default light***, the first solution, turns out to be in any case the most comfortable, quick and elegant.



■ **Figure 3.17**
This time the default light has been deactivated too. Now the scene is not lighted. The render is completely dark. No GI is active in the scene.

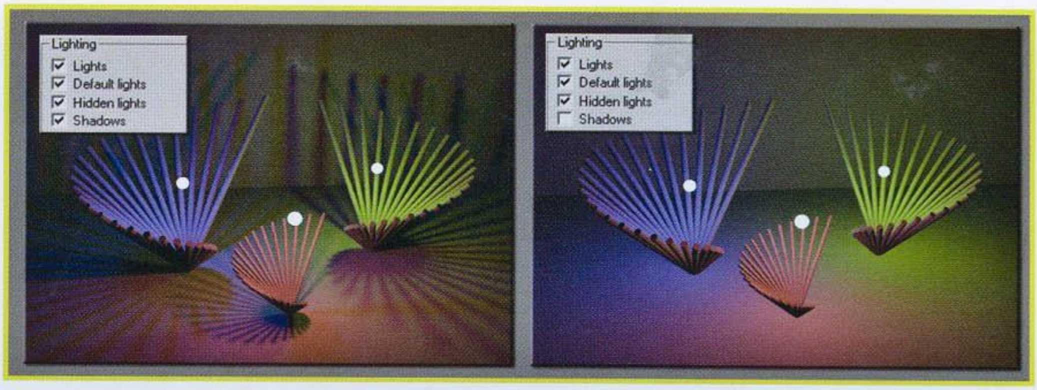


■ **Figure 3.18**
The same situation seen before, but with the GI activated. As it can be noticed, if the 3D models are lighted with ***skylight*** together with the default light, the scene turns out to be overexposed and not correctly lit.

Hidden light - option which activates or deactivates the use of hidden lights, that is to say not visible in viewport. When the parameter ***Hidden light*** is activated, any invisible lights that are turned on will be used in the render. When the option is disabled, any hidden lights will be excluded from computing.

Shadows - parameter which enables or disables the use of shadows.

Show GI only - when this parameter is active, direct light will not be included in the final render. It is to be noticed that the direct light will be computed all the same in GI computing. But only indirect light will be used in the final render.

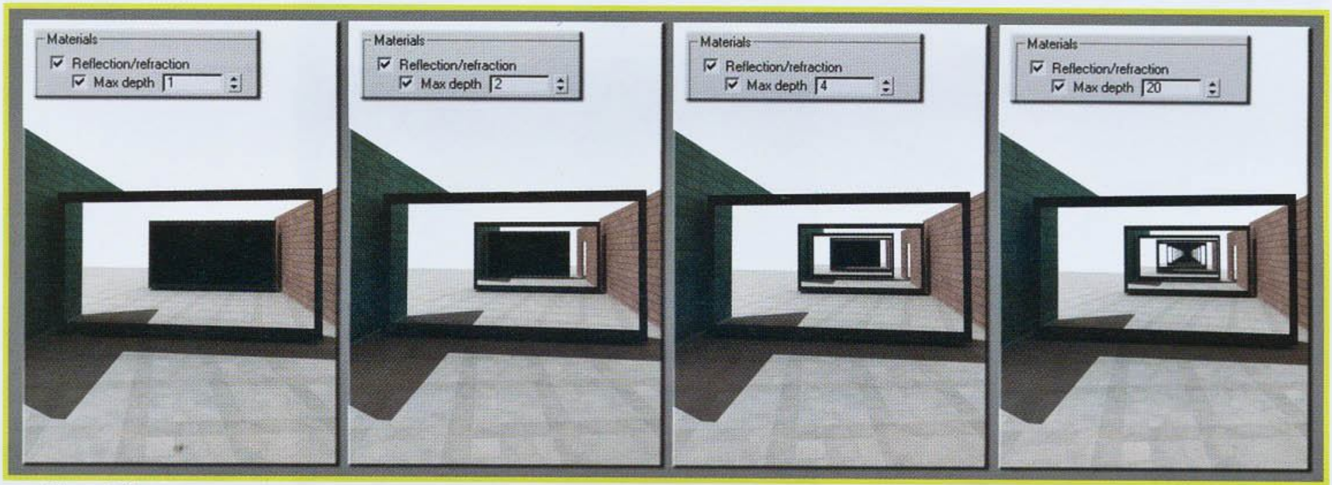


■ **Figure 3.19**
Deactivating and activating the shadows globally makes work faster, and the render too. In this case, when you deactivate the heavy ***Area Shadows***, the render is 4 times as fast. Useful for test rendering.

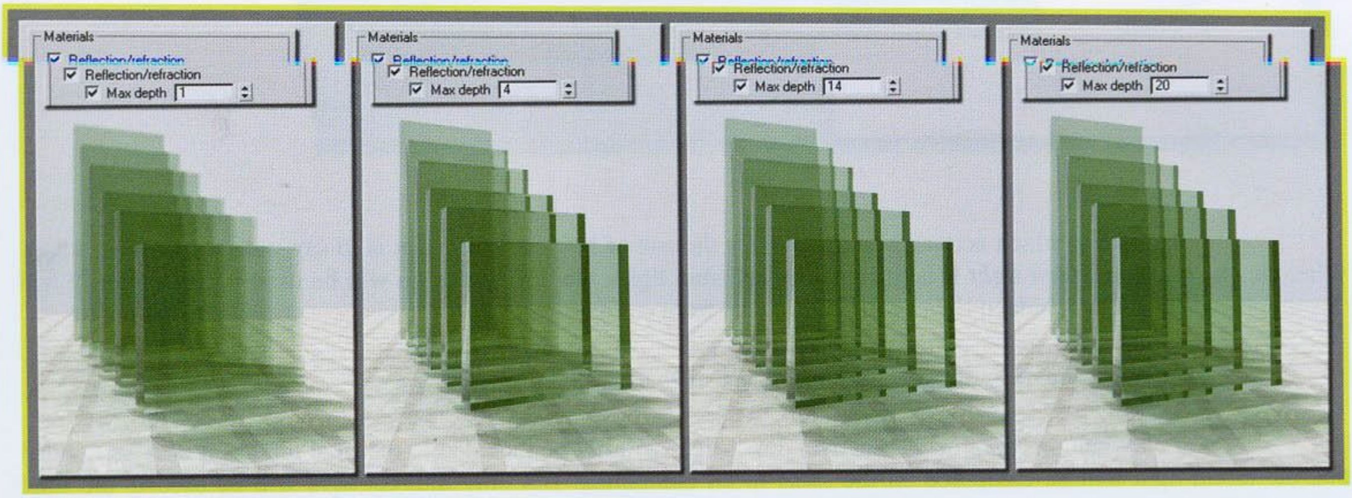
Materials

Reflection/refraction - parameter which globally activates or deactivates the computing of reflections and refractions.

Max depth - parameter allowing to modify globally the computing depth of reflections and refractions. When it has been deactivated, the depth is locally controlled with an option present in the single materials of *VRay*. If activated all the materials will use the depth specified with the parameter ***Max depth***.

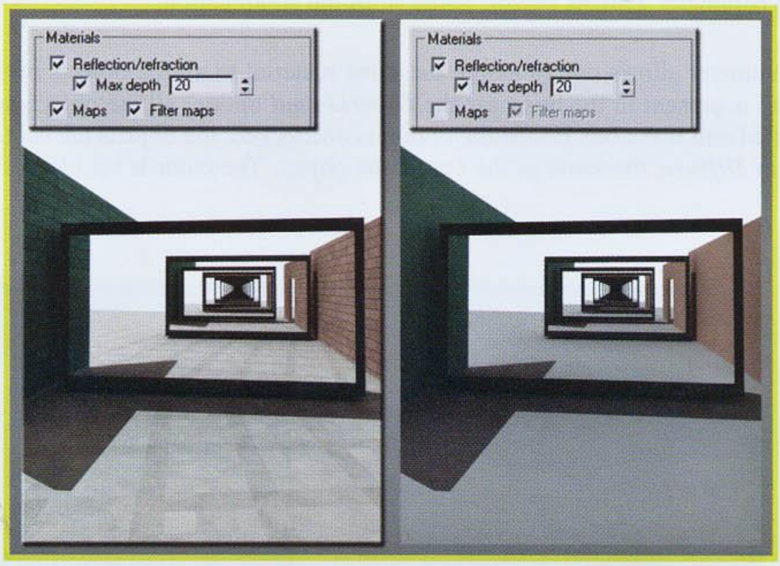


■ **Figure 3.20**
Reflection: image made up of two mirrors placed one in front of the other. The higher the ***Max depth*** value is, the more the number of reflections increases.



■ **Figure 3.21**
Refraction: image made up of 6 glass plates. Increasing ***Max depth***, also the refractions computed by VRay increase. In this case, beyond a certain threshold (in this case 14), VRay has already computed all the rays needed for refractions and increasing ***Max depth*** further does not bring any change.

Maps - option which activates or deactivates the employment of any map in materials present on the scene.



■ **Figure 3.22**
If the use maps is disabled, VRay temporarily deletes all the textures, both procedural or not, from the render.

Filter maps - option which activates or deactivates the filtering of the textures. When activated, each texture uses the local settings within the Material Editor. When not, all types of filtering are disabled.
Filters, which are located in the Material Editor, are used to filter image files principally act to texturize 3D objects. Within the Filtering section, inside the Bitmap Parameter rollout, two typologies of filters are present: Pyramidal and Summed Area. The main difference is the following: Summed Area applies a softer filtering, similar to the ***antialiasing Area***, while Pyramidal is more intense, similar to the ***antialiasing Catmull-Rom***.

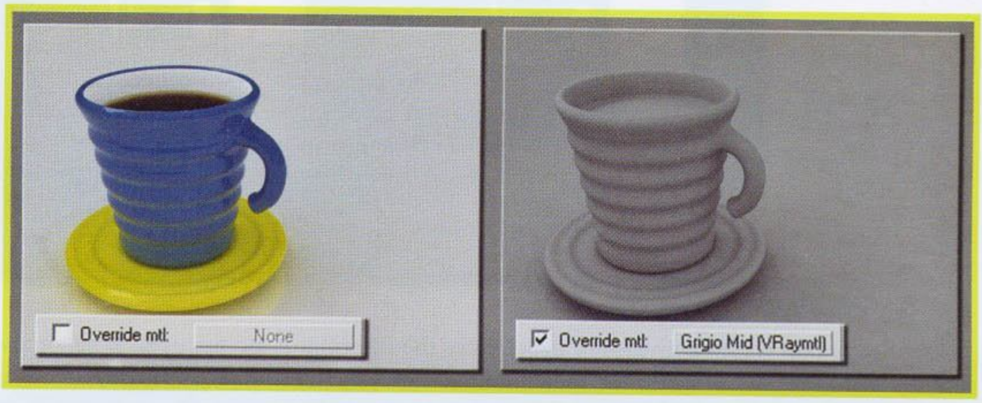
■ **Figure 3.23**
VRay automatically deletes the filtering of textures when the option ***Filter maps*** is deactivated, showing each single texel. As far as high resolution images are concerned, this effect could be ignored, but this is not true for low resolution textures. The little square shows the complete sample of the considered texture.

Max. trans levels - parameter controlling the depth of the use of transparent objects. It shows the maximum level of transparency of an object when light passes through it.

■ **Figure 3.24**
This parameter affects the behaviour of transparent or semi-transparent objects. Notice the effect on the shadows.

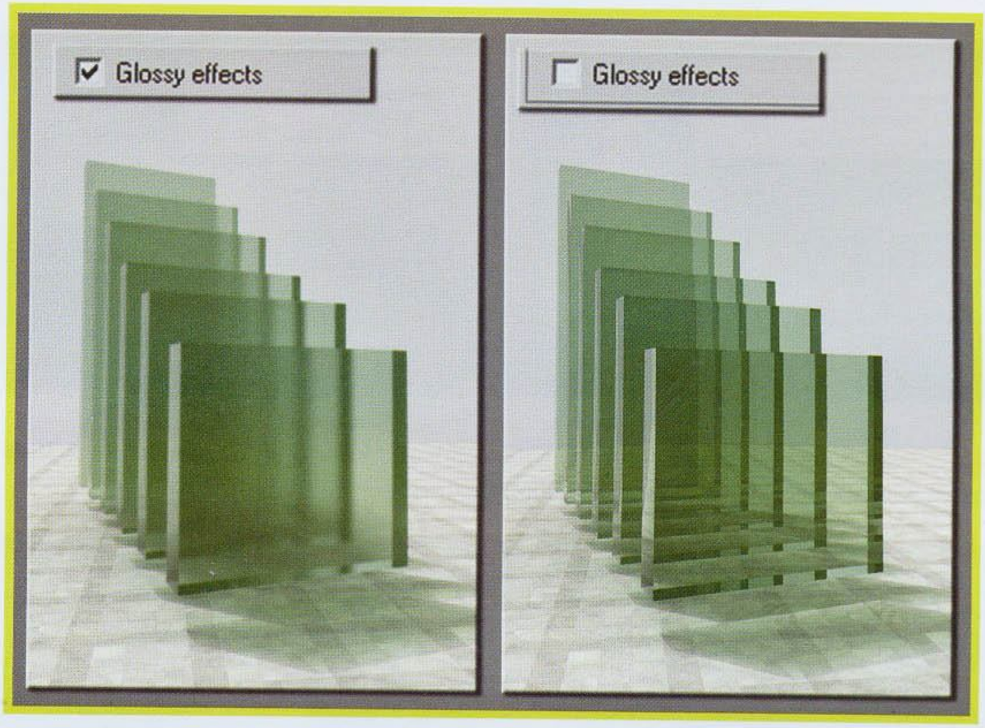
Trasp. cutoff - this parameter checks that the number of rays passing through transparent objects is enough for a certain level of transparency; all the rest of the rays are stopped. If transparency cumulated from rays is lower than the threshold indicated by the parameter ***Transp. cutoff***, *VRay* stops the computation.

Override mtl - this parameter allows one to assign the same material to all the objects present on the scene. When activated and if any shader is present in the box near the ***Override mtl*** option, all the objects acquire the shader which has been set. When activated and if the box beside the option is unchecked, the objects are set to a default shader. It is a *VRay* shader with the color ***Diffuse***, the same as the one of the object. The color is visible in the square near the name of the object (wirecolor).



■ **Figure 3.25**
Clicking on the button ***None*** it is possible to choose one of the 3ds Max shaders. The selected shader is temporarily applied to all the objects in the scene.

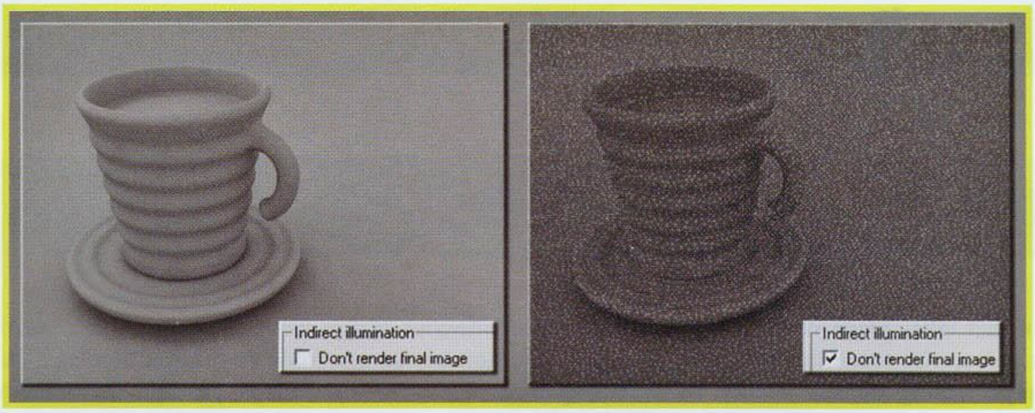
Glossy effects - this parameter globally activates or deactivates the faded reflections of the shaders. The blurry reflections, also called ***glossy***, need a lot of resources and thus time. Their deactivation is useful when quick tests must be carried out.



■ **Figure 3.26**
The ***glossy*** option contained in the shader of VRay, enables blurry reflections and refractions. Thanks to the option ***Glossy effects***, the effect of blurry reflections can be activated or deactivated globally on all the shaders present on the scene.

Indirect illumination

Don't render final image - when Global illumination is used (***Photon Maps, Light Cache, Irradiance Maps***), if this option is activated, *VRay* computes the GI only and leaves out the final render, the real image rebuilding process. Thus reflections, refractions, ***antialiasing*** etc.... will be omitted. It is useful for computing GI maps in case of animations in ***Fly-through*** mode.



■ Figure 3.27
Enabling the ***Don't render final image*** option, VRay only computes the GI, leaving out what should usually happen. Effects such as the computation of ***Area Shadows***, ***glossiness***, ***antialiasing*** etc... are not processed. This parameter will be discussed later, when we cover the subject of ***Fly-through*** animations.

Raytracing

Secondary rays bias - it is a parameter representing a virtual displacement of the polygons of a 3D object, therefore it is not physically visible in the geometry. This displacement is strictly applied in secondary ray computation and during the rendering process. A ***Secondary rays bias*** greater than zero can be useful when the render shows dark stains, due to perfectly overlapping polygons. Usually this kind of problems occurs when importing meshes from programs external to 3ds Max or using objects with overlapping faces.

■ Figure 3.28
On the left every object is composed by two perfectly adjoining texturized levels. Transparency is due to an ***alpha*** channel application to the level. In order to avoid artefacts during the rendering process the parameter ***Secondary bias*** has been increased and set to values greater than zero.

VRay: Image sampler (Antialiasing)

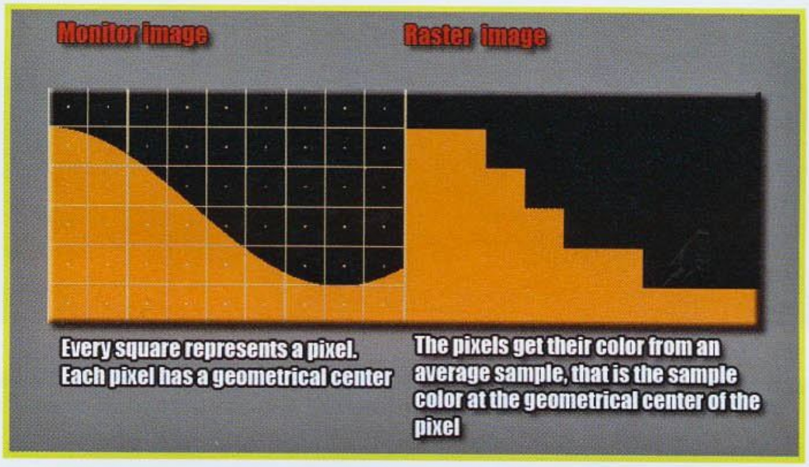
INTRODUCTION

Aliasing is a phenomenon that takes place when one tries to build a continuous image using discrete values, that is when one wants to render a vectorial scene through raster images. Unfortunately an unavoidable information loss, due to the different way of representing objects occurs, as a consequence of the vectorial-raster passage. In order to minimize the problem of pixel use, that is, the typical "jaggedness" of image edges, very high rendering resolutions could be used: thanks to a higher resolution a table edge could then be less jagged. Of course this is a good way to handle the problem, but it involves a greater demand of hardware and time resources. In any case, ***aliasing*** occurs using very high resolution as well.

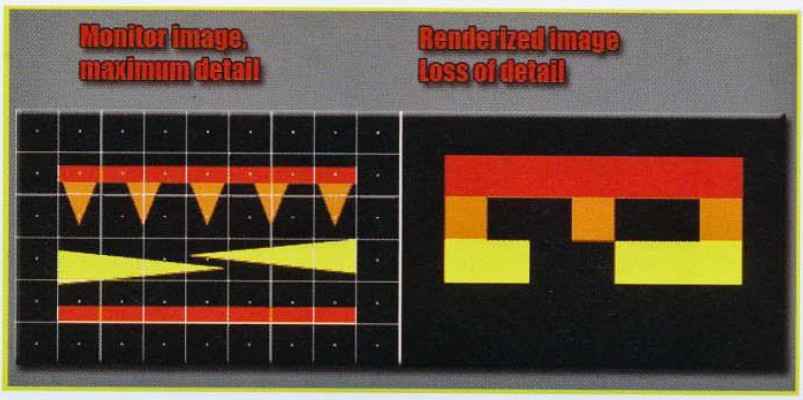
There are two main ***aliasing*** types:

Spatial aliasing: when artefacts are considered in the single static image;
Time aliasing: when artefacts are considered in a sequence of images (animations). This problem leads to shaking or flashing objects. Sometimes it might occur in particular situations, such as the animation of a wheel turning in the opposite way compared to the direction it is moving in. This problem is due to the fact that small objects appear in some frames but not in some others: this is the typical phenomenon coming from an incorrect ***antialiasing***.

As everybody knows 3ds Max works with primitives, triangles and lines: these are adjoining objects, because of their vectorial nature. They turn to a pixels ensemble during the raster only. This turning is named **sampling** and it yields a visual artefact called **aliasing**. During the sampling process the color of a pixel is fixed referring to the centre of the pixel itself. The central point of the pixel is taken from the geometry lying behind the pixel grid fixed by the output resolution.



■ **Figure 3.29**
Aliasing is a phenomenon that takes place trying to build a continuous image through the use of discrete values: information loss occurs, and this leads to poor image quality and to jaggedness.

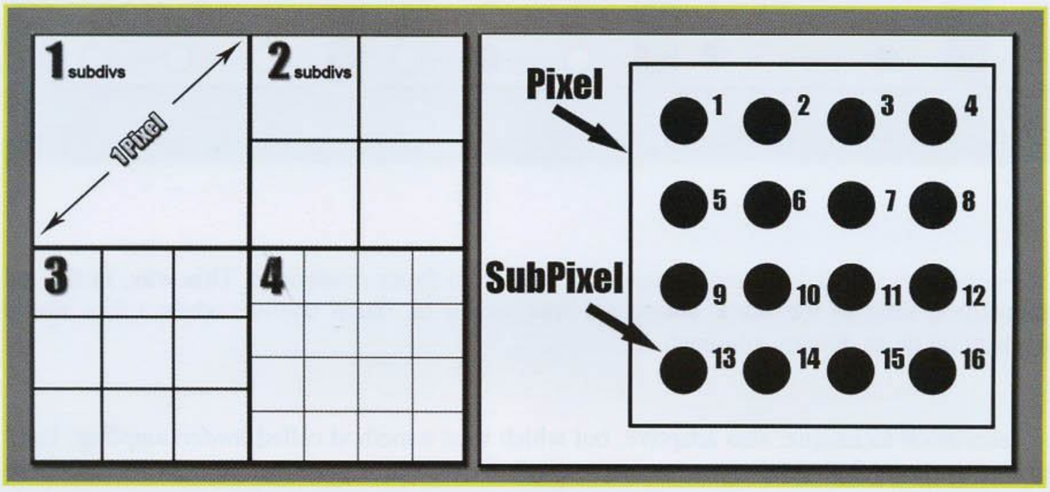


■ **Figure 3.30**
A low resolution image. After the rasterization the image appearing in the monitor is unintelligible. Every square is a pixel that takes the color from the vectorial geometry lying behind. It takes the color corresponding to the pixel's geometric centre, represented by the white dot.

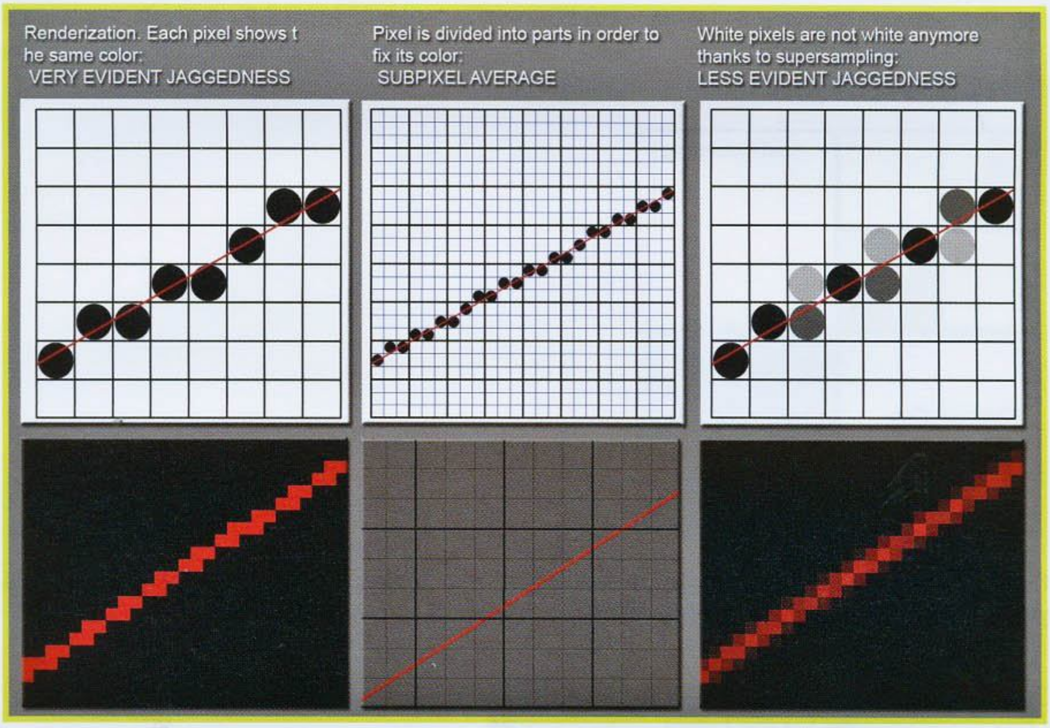
The aim of the ***antialiasing*** technique is to improve image quality by reducing or removing artefacts due to ***aliasing***.

There are a several techniques for the computation of ***antialiasing***. One is called ***oversampling***, also called ***supersampling***.

Through ***supersampling*** each pixel is divided into ***subpixels***. The color of each ***subpixel*** is separately computed. To obtain the final color of a pixel it is necessary to compute the average of the colors of each ***subpixel***. The ensemble of ***subpixels*** is called ***subpixels mask*** (e.g. 2x2 = 4 ***subpixels***, 4x4 = 16 ***subpixels***). ***Antialiasing*** quality is higher, the higher the number of ***subpixels*** for each pixel. A 4x4 mask is supposed to lead to a very good visual quality, but requires a large raster time.



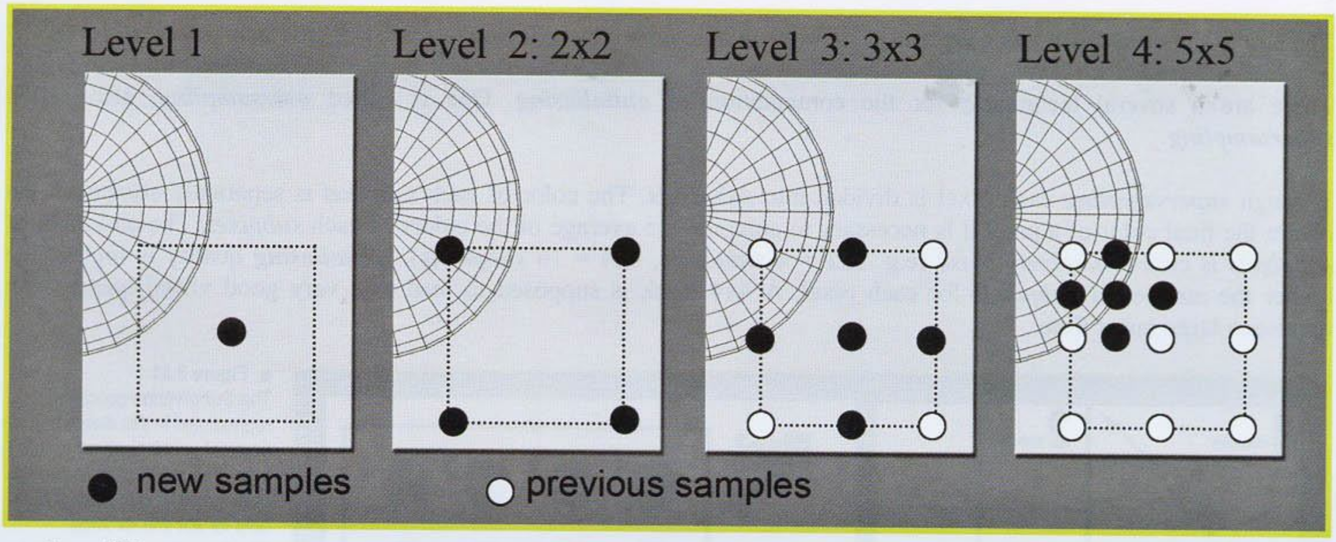
■ Figure 3.31
The ***Subdivision*** concept is represented in this diagram. It is a frequent parameter among VRay's tools. The subdivision represents a pixel subdivision in smaller parts, both for ***AA*** and for ***Area Shadows*** etc… The real number of subdivisions is given by the square of input value. ***Subpixels*** scheme.



■ Figure 3.32
Vectorial line rendered before and after ***antialiasing***.

In the upper example, all the pixels of the render are divided in subpixels using a 3x3 ***subpixel*** mask, even those that do not need it, such as totally black ones or the ones without any vectorial data. In this case it is possible to avoid the subpixel division and therefore allow the AA calculation to be faster. This is why the idea of adaptability has been introduced. Based on a threshold fixed by the user, *VRay* seeks out the pixels of the image which require ***antialiasing***. After, these pixels are divided in ***subpixels***.

VRay tries to find the ***subpixels*** that need another subdivision, skipping the sufficiently detailed ones, according to the threshold fixed by the user. This process continues until the maximum subdivision number set has been reached.

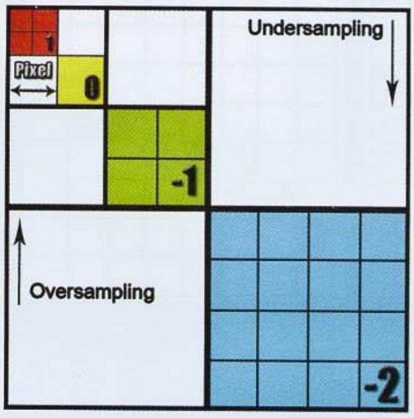


■ **Figure 3.33**
System of ***Adaptive sampling***.

Working only on strictly required area allows less computations and therefore faster renderings. This way, in the red segment example, ***AA*** computation time in the black undetailed area would be much shorter, while *VRay* would exclusively focus on red pixels or on those directly adjoined.

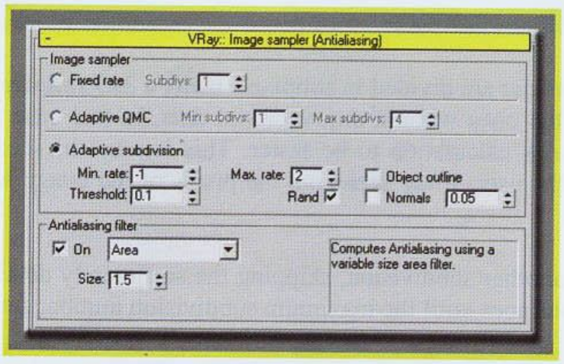
VRay provides another ***AA*** calculation technique, also adaptive, but which uses a method called ***undersampling***. Until now the pixel was divided into parts. A 2x2 subdivision means a pixel division in 4 parts; while a 5x5 subdivision means 25 subpixels, without adaptability, and so on. The ***undersampling*** works in the opposite direction: one sample is used for more than one pixel. For instance, applying a -1 ***undersampling*** value, 1 sample is used for 4 pixels, applying a -2 ***undersampling*** value 1 sample is used for 16 pixels etc… Plainly this method leads to excellent quality/time results in the case of an undetailed scene or one presenting very smooth surfaces

■ **Figure 3.34**
Undersampling is not just used for ***AA***, but for ***Irradiance Map*** and for ***glossy*** effects of materials as well.



VRay provides 3 types of algorithms for ***antialiasing*** computation: ***Fixed rate sampler***, ***Adaptive QMC sampler*** and ***Adaptive subdivision sampler***. Moreover almost all 3ds Max filters are supported.

■ **Figure 3.35**
On this board it is possible to choose the sampling type, in order to avoid ***aliasing*** on the rendering.



PARAMETERS

Image sampler

Fixed rate sampler

It is the easiest sampling method. It computes the ***AA*** of each individual pixel, according to the number of ***subpixels*** set. It globally performs a subdivision of **all** pixels in the image, without any adaptability. Regardless of geometrical detail, each pixel is always divided into the same number of ***subpixels***.

Subdivs - it fixes the number of samples for each pixel. If this value is 1, then only one sample is considered for each pixel. If the parameter ***Subdivs*** is greater than 1, then each pixel is divided by the square of the set value. It represents an ***AA*** direct-calculation method. It is advisable to use this ***AA*** model when working with images full of effects, such as ***Area Shadows***, ***SSS***, ***Motion blur***, ***DOF*** and very complex textures.

Adaptive QMC sampler

This algorithm works on a varying number of samples for each pixel, according to the intensity difference among the neighbouring ones. Like all blurry effects, this method is closely influenced and related to the parameters in Rollout ***VRay: rQMC sampler***, especially the ***Adaptive amount*** and ***Noise threshold*** parameters. The first of these determines how much the sampler is adaptable: adaptability is greater when this parameter is higher. The second one deals with choosing which ***subpixels*** must be divided. It sets the threshold of quality within which ***VRay*** establishes if a particular ***subpixel*** is useful or not for ***antialiasing*** computation: the quality is higher as the value is lower and vice versa. It is recommended not to use values lower than 0.002. Values which are too low result in a very long rendering time.
It is preferable to apply the ***Adaptive QMC sampler*** to very detailed images, like ***VRayFur***, ***glossy***, ***Depth Of Field*** (***DOF***), ***Motion blur*** etc… Moreover, this sampling uses less RAM than the next method: the ***Adaptive subdivision sampler***.

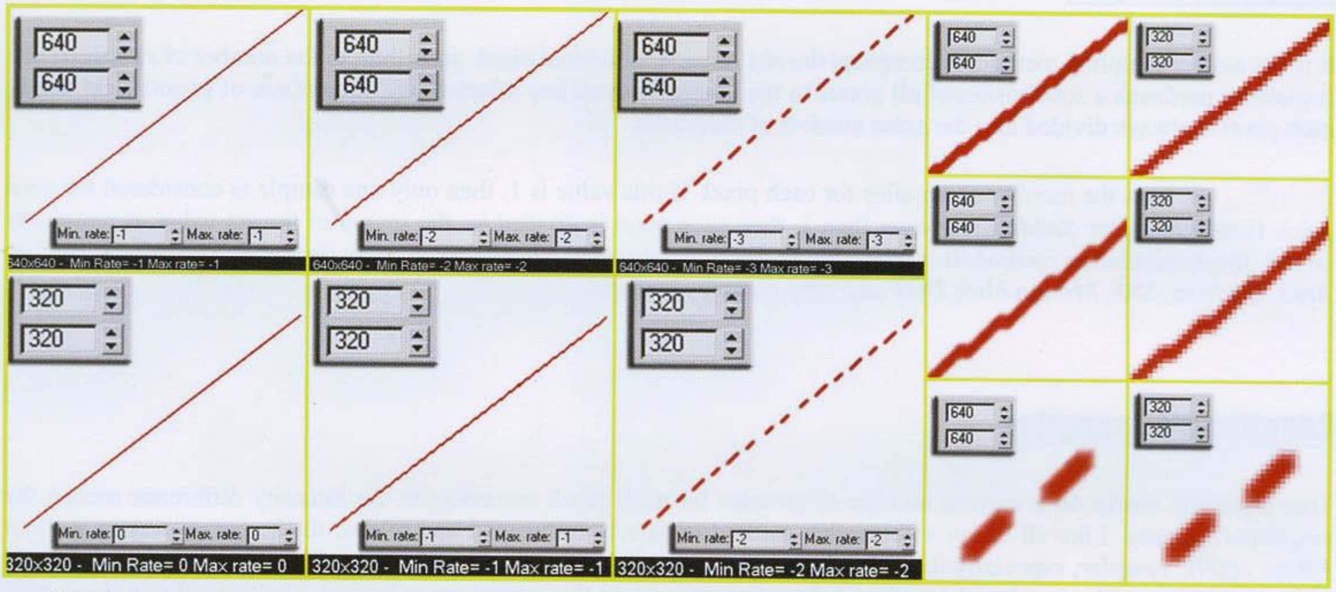
Min subdivs - it fixes the minimum sample-number to be used in each subdivision of pixels. The higher its value, the more samples are considered. Therefore quality is higher at the cost of penalizing rendering time. This parameter sets the minimum number of subdivisions per pixel below which ***VRay*** is not allowed to go. Obviously the minimum value is 1. Increasing its value, e.g. ***Subdivs*** = 2, ***VRay*** uses at least 4 ***subpixels*** for each pixel to perform the ***antialiasing*** computation.

Max subdivs - it fixes the maximum number of samples used for each pixel. The higher it is, the more subdivisions are created, if necessary, during the sampling process.
Previously we mentioned that ***VRay*** proceeds with the subdivision of pixels in ***subpixels*** according to the threshold set by the user. Which is the initial subdivision limit? It is fixed by the ***Max subdivs*** parameter. Usually the value 4 is sufficient in most cases. However sometimes values greater than 4 are needed, as is the case when very thin lines or noise problems due to too low ***Subdivision***.

Adaptive subdivision sampler

It is an advanced ***AA*** computation method exploiting an ***undersampling*** system. Basically, for a given resolution, it is as if *VRay* renders to a lower one returning the render with the original resolution once the ***AA*** computation is finished. Therefore it might happen that a sample is used for more than one pixel in some render areas. The following examples will clarify this concept.

A simple line, rendered at a 640x640 resolution, has been created. Using ***antialiasing Adaptive subdivision sampler*** both ***Min rate*** and ***Max rate*** parameters have been set to -1, which is the first ***undersampling*** level. Then ***undersampling*** values of -2 and -3 have been applied. At last the resolution has been modified, reduced by half exactly. This time ***undersampling*** values of 0, -1, -2 respectively have been used.



■ **Figure 3.36**
Antialiasing Adaptive subdivision sampler example applied to a real situation.

The image above shows how ***antialiasing*** does not change, even if applied to different resolutions. There are some problems due to the low resolution of 320x320-pixel-renders, however this does not affect the demonstration of this concept. The jagged edges, which in this case are intentionally macroscopic, are completely equal in both the resolutions. This means the calculation resolution is actually reduced when using ***undersampling***. The ***undersampling*** concept will be found again in the ***Irradiance Map*** calculation and in the subject of unfocused reflections and refractions.
Without any blurry effects (***direct GI, DOF, glossy***, etc...) it is better to apply this method instead of the ***Adaptive QMC sampler*** method. Generally the ***Adaptive subdivision sampler*** algorithm uses less samples to perform the ***AA*** computing compared with the other two algorithms, therefore taking less time to reach the same quality. However, when detailed structures are involved, this ***antialiasing*** type might produce worse quality images in a longer time. Moreover it requires much more RAM.

Min Rate - this parameter controls the minimum number of samples for each pixel. Setting a value equal to zero *VRay* uses one sample for each pixel. Setting -1 one sample for 4 pixels is used, -2 means one sample for 16 pixels.

Max Rate - this parameter controls the maximum number of samples for each pixel. Setting a value equal to zero *VRay* uses one sample for each pixel, 1 means four samples for each pixel and 2 means sixteen samples for each pixel.
Some softwares represent both the ***Min Rate*** and the ***Max Rate*** as a fraction, and not as an integer. As a consequence:

■ **Figure 3.37**
These fractions might seem familiar to many readers...

-2	1/16	1 sample for 4x4 pixels. One sample defines 16 pixels.
-1	1/4	1 sample for 2x2 pixels.
0	1	1 sample for each pixel.
1	4	4 samples for each pixel.
2	16	16 samples for each pixel.
3	256	256 samples for each pixel.

Threshold - this parameter fixes sample sensitivity. Low values lead to better results; high settings speed up the rendering, but there is the risk of creating artefacts. This parameter affects the ***antialiasing*** of textures applied on objects, as well as on edges.

Rand - this parameter places samples randomly and allows a better ***AA***, especially when there are vertical or horizontal lines.

Object outline - this parameter fixes a ***supersampled*** system for the external edges of rendered objects. In practice it directly enables or disables the ***AA*** on object edges, without adaptability. This parameter does not affect ***DOF*** and ***Motion blur***.

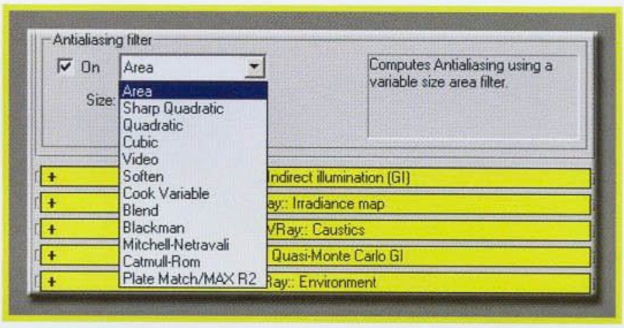
Normals - this parameter oversamples the areas contained within objects which have a high variation of polygon normals. Whereas the parameter ***Object outline*** activates or deactivates the ***AA*** along external edges of the objects, ***Normals*** allows one to decide whether to sample internal corners of the object too. This parameter does not have any effect on the ***DOF*** or ***Motion blur***.

In brief, ***Color Threshold***, ***Object outline*** and ***Normals*** are additional parameters which allow one to choose where to apply ***antialiasing***. Later the use of these parameters will be visually clarified with some practical examples.

Antialiasing filter

Antialiasing filter

When ***antialiasing*** and ***subpixels*** subdivision are computed, it is important to establish their contribution to the final render. The filters, which operate at ***subpixel*** level, allow one to define or fade the final output in accordance with the kind of filter set. All 3ds Max standard filters are supported by *VRay*, except for ***Plate Match***.



■ Figure 3.38
VRay uses the same 3ds Max filters.

Area - it computes ***antialiasing*** by using an area filter with variable dimension, which is the original 3ds Max filter.

Sharp Quadratic - it is a 9-pixel rebuilding filter called Nelson Max filter.

Quadratic - it is a 9-pixel blur filter based on quadratic spline.

Cubic - it is a 25-pixel blur filter based on cubic spline.

Video - it is a 25-pixel blur filter optimized for NTSC and PAL video applications.

Soften - it is a Gaussian blending filter adjustable for mild blurring.

Cook Variable - it is a filter for general purpose. Values from 1 up to 2.5 lead to high resolution, while upper values make the image unfocused.

Blend - it is a mixture of defined area and Gaussian blending filters.

Blackman - high definition 25-pixel filter, but without edge optimization.

Mitchell-Netravalli - it is a two parameter filter: blur interchange, ringing and anisotropy.

Catmull-Rom - it is a 25-pixel rebuilding filter characterized by a mild edge optimization effect.

NOTES

Now the question is: which sampling style is preferable to use? The only way to choose is by doing many tests and trying to understand which situations make one method more suitable than another one. In any case, some suggestions might be useful:

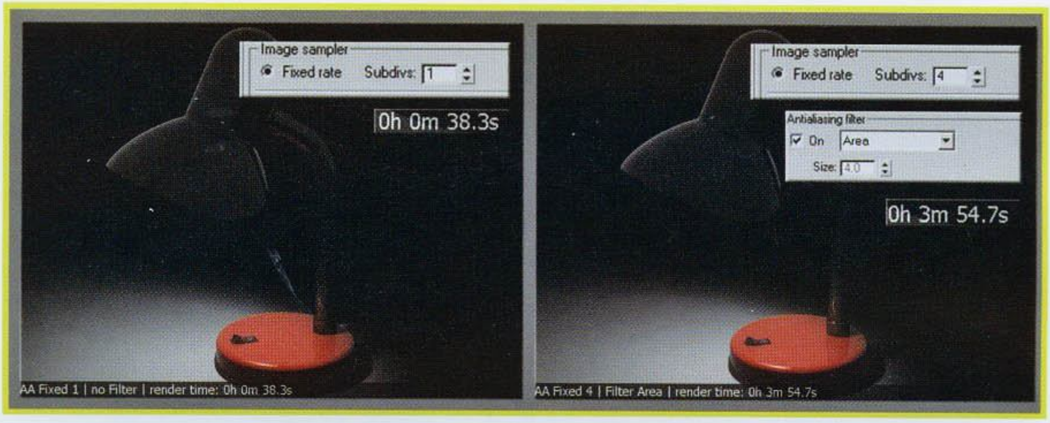
- ***Adaptive subdivision sampler*** is the best solution to be applied in those scenes with few blurry effects and textures poor of details, thanks to its ***undersampling*** capability and speed in low detailed zones of the image.
- ***Adaptive QMC sampler*** offers the best performance in those scenes with very detailed textures with small geometries, and with very detailed animations too.
- ***Fixed rate*** offers the best quality in the case of extremely elaborate scenes, rich of effects, with small objects and detailed textures.
- Any adaptability is deleted if ***Min subdivs*** is equal to ***Max subdivs*** (or ***rate***).

The last recommendation is about RAM use. Image sampling requires a lot of RAM to store information in each rendering panel (***bucket***). Using big size ***buckets***, *VRay* increases the demand of RAM. This may happen when using the ***Adaptive subdivision sampler***. Thus it is worth avoiding its use for heavy scenes requiring a vast amount of memory.

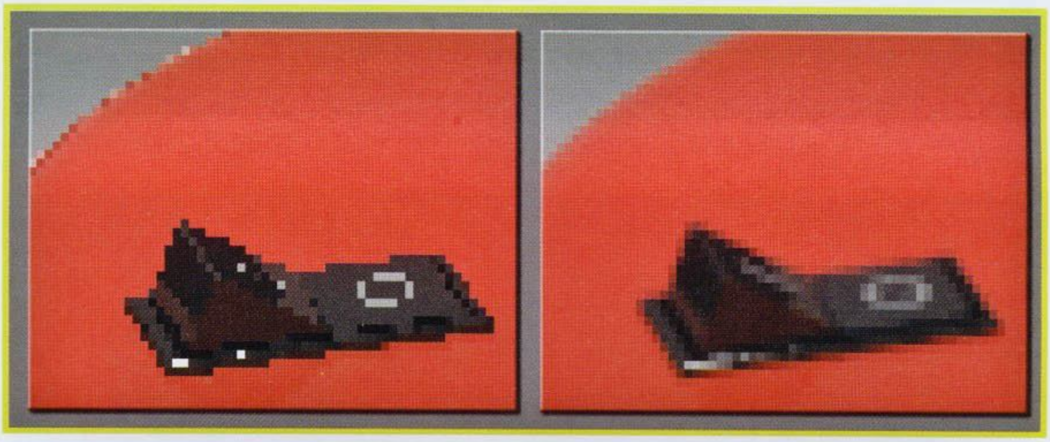
EXMPLES

Example 1: what is antialiasing?

The same image with and without ***antialiasing*** is shown in the following examples.



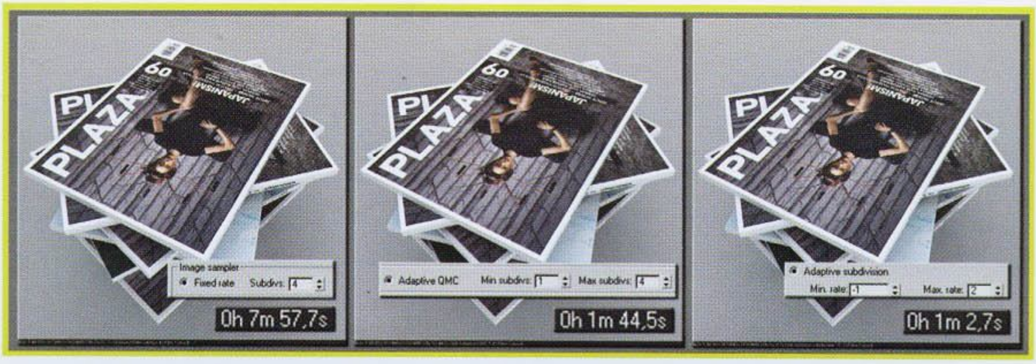
■ **Figure 3.39**
The left image was calculated without ***AA***, applying the ***Fixed rate*** sampling method with ***Subdivision*** set to 1. For the right-hand image the ***AA*** was enabled through the ***Fixed rate*** method as well, but increasing the ***Subdivision*** parameter. Both rendering time and higher quality are visible.



■ **Figure 3.40**
Enlargement of the previous scene. The switch pixels now show a hazy stair-effect instead of a sharp one, as in the left image.

Example 2: Adaptive subdivision sampler

A sequence of rendered scenes is shown in the following examples, where the three different sampling methods are applied: in each case quality and computing time are compared. The following scene is a very simple one: it shows linear geometries, no elaborate texture, no blurry effects, no ***Area Shadows*** etc…

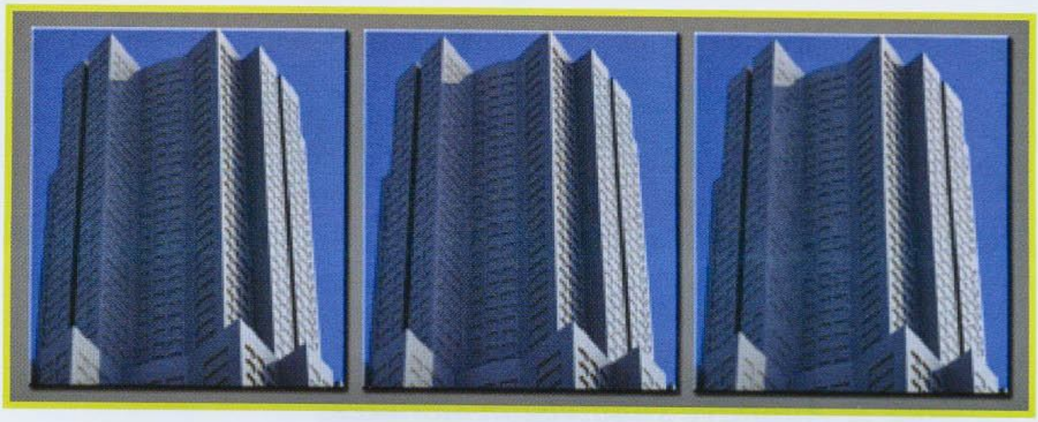


■ **Figure 3.41**
Since no particular blurry effects are present, the ***Adaptive subdivision*** sampling method tends to be the best one for images of equal quality. Instead in this kind of situation, the ***Fixed rate*** method is very slow even in areas without any ***antialiasing***, due to its non-adaptability.

Example 3: Adaptive QMC sampler

We know that each sampling method has some merits and some faults. Previously it was shown that the ***Adaptive subdivision*** is the best way to handle situations with very few blurry effects. But now we will see a scene where ***Adaptive subdivision*** can lead to long computing times and unsatisfying quality, even without particularly complex effects. This is because the ***Adaptive QMC sampler*** is the best solution where high precision is needed (geometrical grids, models with very thin zone). Especially when *QMC+QMC* is used as GI computing method, as we will see.

■ **Figure 3.42**
This scene has a lot of small details on the windows. That is why the ***Adaptive QMC*** is the best method in terms of quality/time ratio. Using the ***Adaptive subdivision*** method, quality is definitely lower, even with triple rendering time, especially in the uppermost part of the tower, where ***antialiasing*** has failed in many ***pixels***.
--- DVD ---



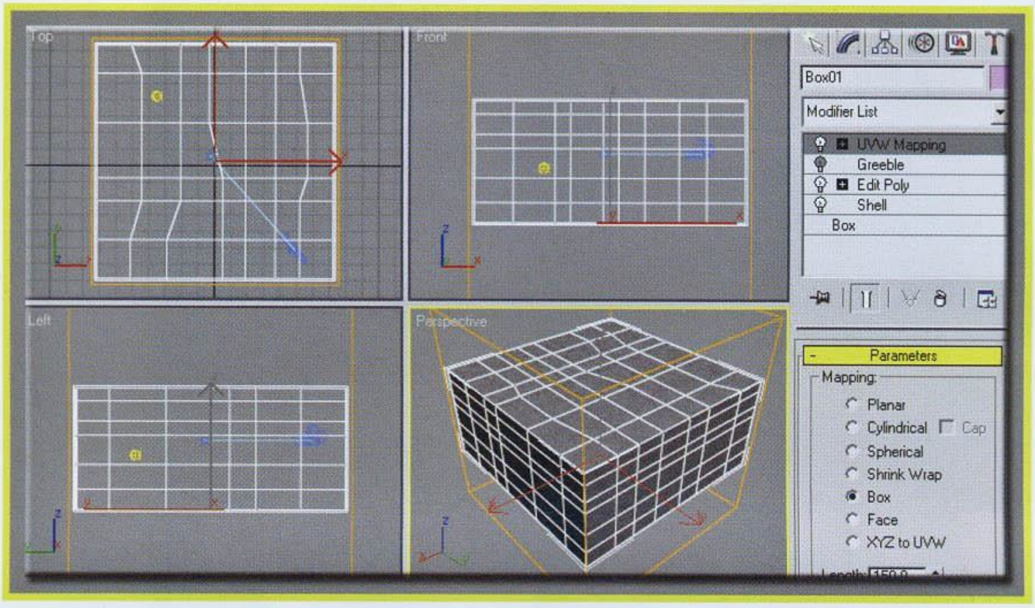
■ **Figure 3.43**
Scene enlargement.

In order to improve the quality of the rendered image using ***Adaptive subdivision sampler,*** one could modify ***Min rate***, ***Max rate*** or ***threshold*** reducing it to 0.01. In this way the problem of "lost" pixels could be solved, but rendering time would increase considerably.

Example 4: DOF, bump and textures

The next scene is very simple.

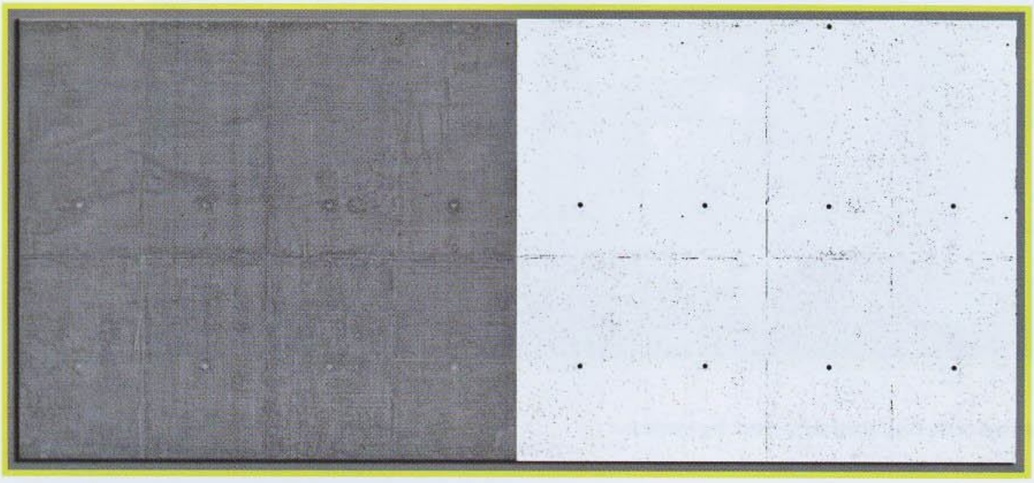
1. Create a 150x150x60 box in Generic Unit.
2. Apply a Shell modifier with Outer value set to 1.
3. Apply an Edit Poly and, using the Vertex mode, move some vertexes in order to create a certain variation in polygon size.
4. Apply a Greeble modifier, which is contained in the DVD. Then modify the parameters, which are simple and intuitive.



■ **Figure 3.44**
Working Area.

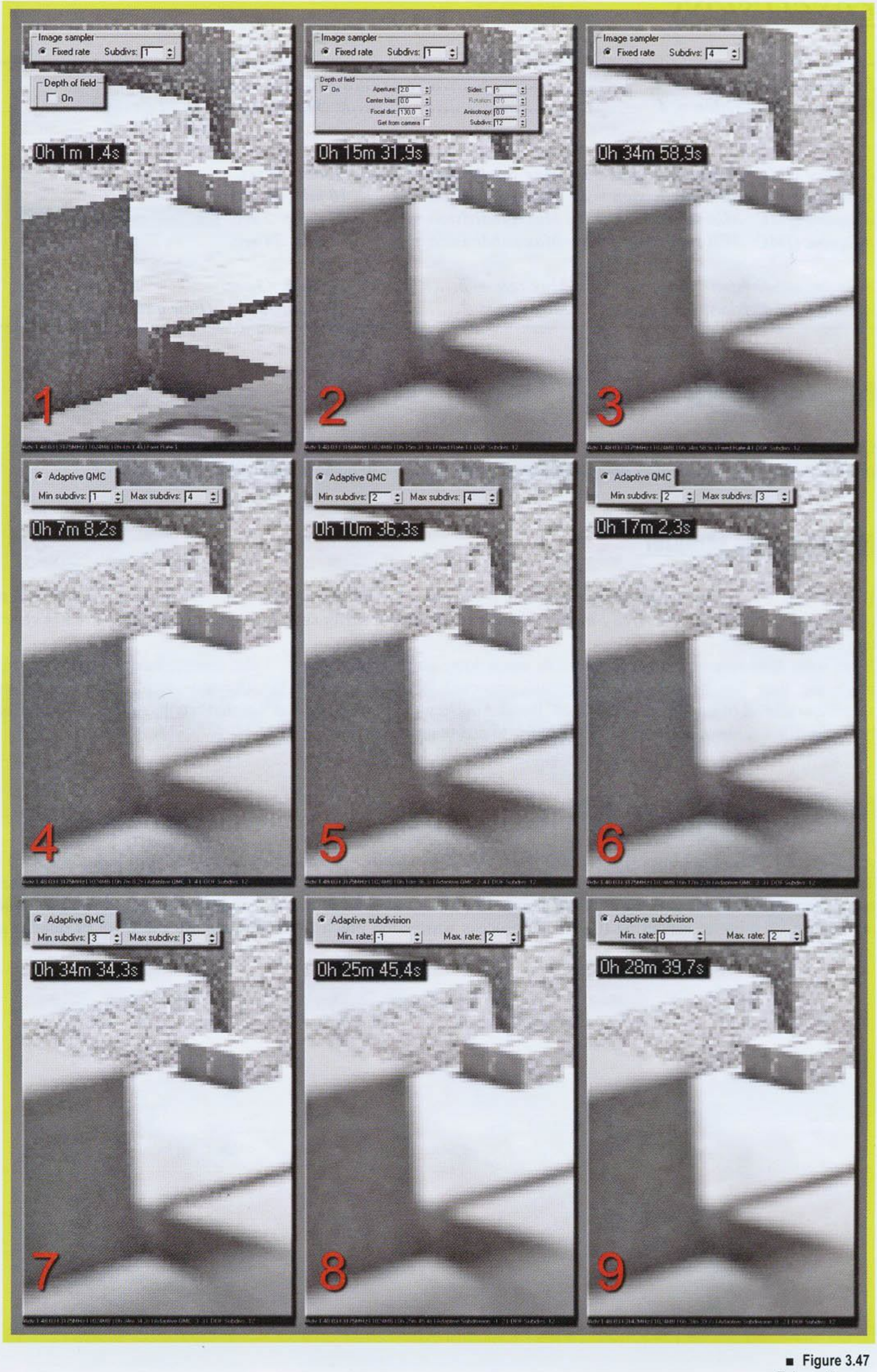
5. Apply ***VRayMtl*** material to the box. Use two maps inside it, one as texture in the ***Diffuse*** channel and the other one as ***Bump***. In this particular case a cement texture with its related ***Bump*** channel, produced by the same ***Diffuse*** map, was applied. Modify ***Bump*** value from 30 up to 100, in order to increase the ***Bump*** effect.
6. A spherical ***VRayLight*** with ***U Size*** = 2.5 and ***Subdivision*** = 12 produces the lighting.
7. To make this test more interesting, the ***DOF*** was also enabled by modifying the parameter ***Subdivision*** = 12. The ***VRayLight*** and ***Subdivision*** correspondence is purely accidental, since 12 guarantees both a good ***VRayLight*** shadows quality and a good ***DOF*** quality.

It is then appropriate to define this scene "complex", because of the presence of a GI, the ***Area Shadows***, ***DOF***, ***Bump*** and complex textures.



■ **Figure 3.45**
Two textures used in the following test. Each map's original resolution is 5200x4600 pixels.

■ **Figure 3.46**
Rendering test with variation of settings of the three antialiasing computing methods.
--- DVD ---



■ **Figure 3.47**
Enlarged image.

CONCLUSIONS

Fixed rate Subdivision = 1		....	*15 min. 32 sec.*
Fixed rate Subdivision = 4		....	*34 min. 58 sec. (Best quality)*
Adaptive QMC	*Min subdivision = 1*	*Max subdivision = 4......*	*7 min. 8 sec.*
Adaptive QMC	*Min subdivision = 2*	*Max subdivision = 4......*	*10 min. 36 sec.*
Adaptive QMC	*Min subdivision = 2*	*Max subdivision = 3......*	*17 min. 2 sec. (Best quality / time ratio)*
Adaptive QMC	*Min subdivision = 3*	*Max subdivision = 3......*	*34 min. 34 sec.*
Adaptive subdivision	*Min rate = -1*	*Max rate = 2..............*	*25 min. 45 sec.*
Adaptive subdivision	*Min rate = 0*	*Max rate = 2..............*	*28 min. 39 sec. (Fair quality)*

Fixed rate sampler

Quality increases when increasing the ***Subdivision*** number, while the internal ***DOF*** noise and the ***aliasing*** presence in textures drastically decrease. An excellent quality can be achieved with ***Subdivision*** = 4.

Adaptive QMC sampler

By applying the default settings rendering time decreases a lot, but the quality is not the highest possible. There is a lot of noise in the ***DOF*** and the ***Area Shadows***. The textures do not suffer such low values, since they are not very complex and rich of details.

Even with ***Min rate*** = 2 there is a visible noise loss. Setting ***Max rate*** = 3 the correct balance between quality and rendering time has been found. Anyway one could be satisfied with the previous step, where in 10 minutes a satisfactory level of quality was achieved. Render noise would be higher, but not that much. Actually this "effect" could be desired, since nowadays there is a tendency to create grainy images.

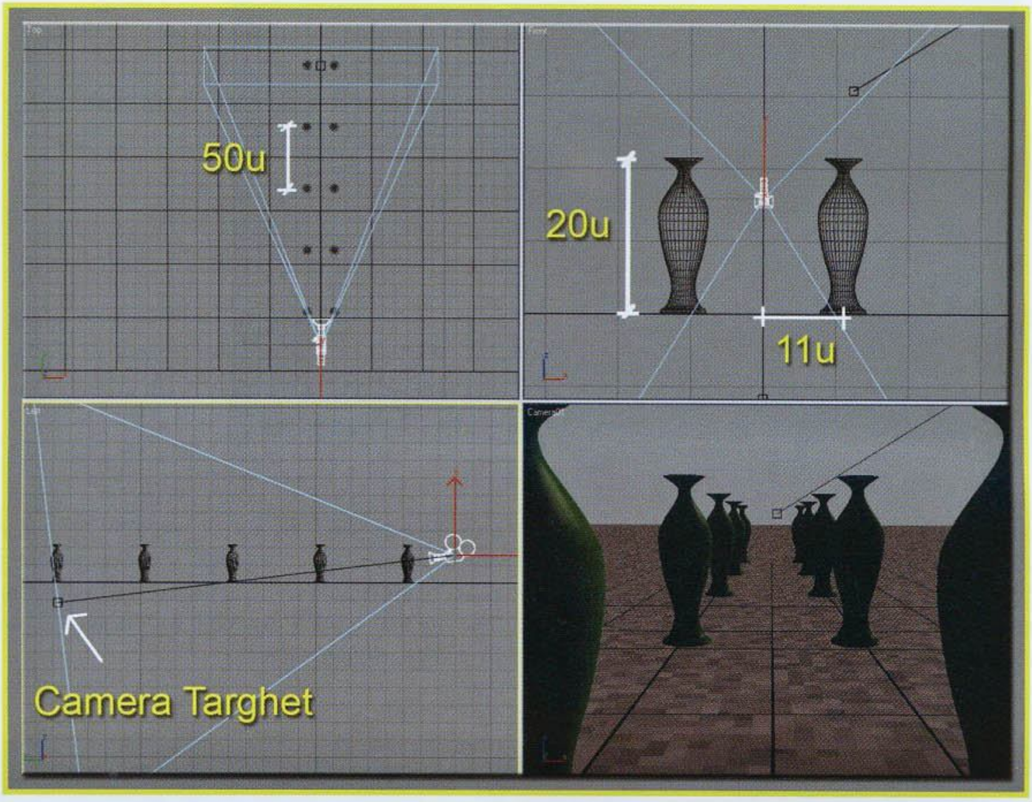
Adaptive subdivision sampler

Applying the default settings ***Min subdivs*** = -1 and ***Max subdiv*** = 2 rendering times are very high: about 26 minutes. The quality is decent, but in the more difficult zones, meaning where ***DOF*** and ***Area Shadows*** are present, these parameters lead to a render with spotted noise. In order to avoid this effect it is necessary to apply ***Min subdivs*** = 0, attaining a 28 minutes rendering time. This process wastes too much time compared to the level of quality achieved.

Example 5: DOF and procedural textures

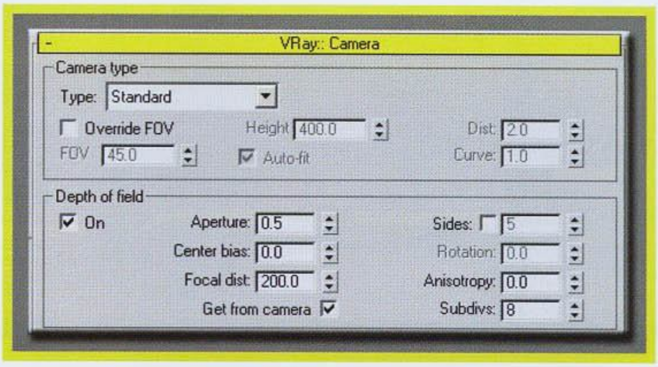
The next scene is very simple:

1. Create a plane with rather high dimensions, for instance 2,600 x 1,600 ***Generic Unit***. Once a Tiles texture has been applied, this allows one to test the different ***antialiasing*** methods, because of the wide surface to be rendered. Place the plane in the centre of the scene.
2. Create a couple of objects about 20 units high, -11 and 11 units away from the centre. Generate a series of 5 objects at 50 units distance from each other.
3. Create a camera with the ***target*** positioned upward in the Top view, with coordinates 0, -1300.
4. Align the camera ***target*** with the last object of the scene. This allows one to set the ***DOF*** very quickly.



■ **Figure 3.48**
3ds Max software view. The scene is composed by 10 polygonal vessels, a camera and a lighting generated by ***VRaySun + VRaySky v1.47.03***. The aim is to test all the ***antialiasing*** methods regarding the quality/time ratio.

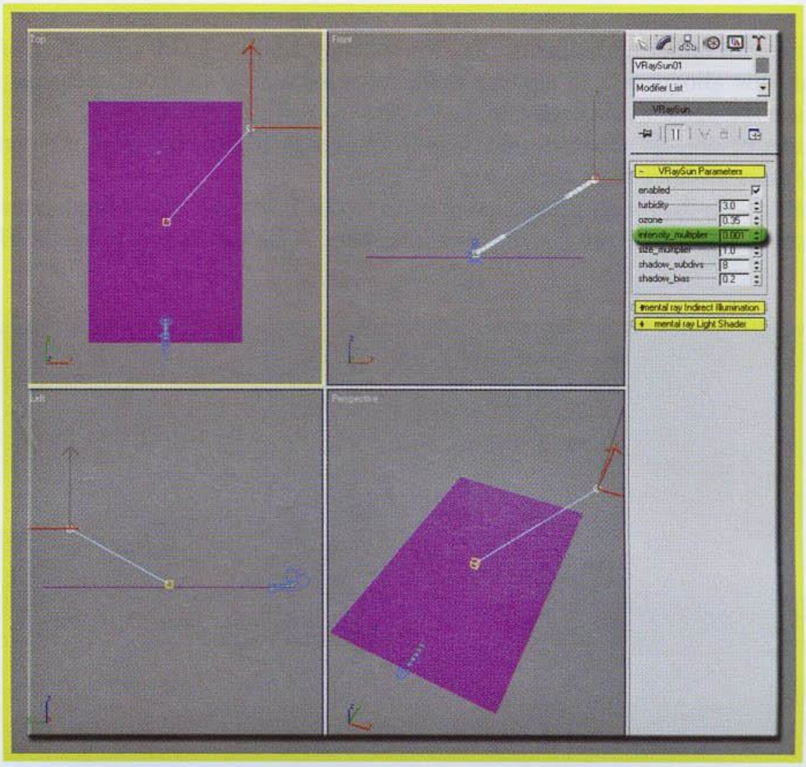
5. Set the ***DOF*** as in the picture. Modify the ***DOF Subdivs*** parameter from 6 to 8, in order to obtain higher rendering quality.



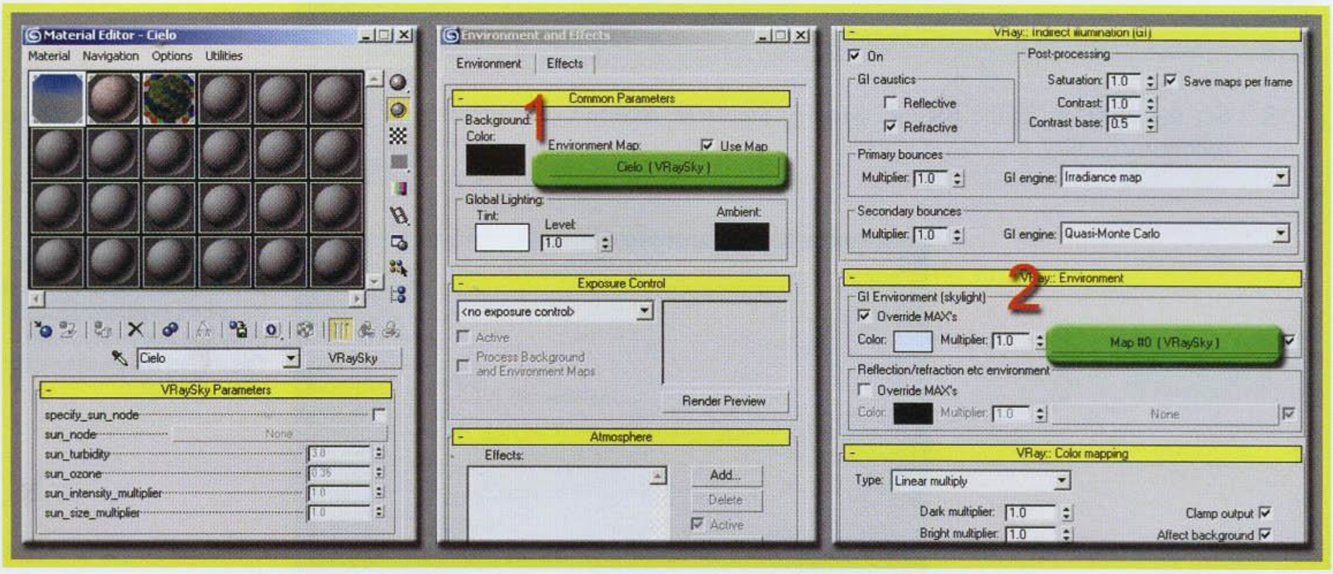
■ **Figure 3.49**
Enabling the ***Get from camera*** parameter, the objects aligned with the camera ***target*** are focused. Everything else is not focused. This is a fast and precise method in the ***DOF*** setting.

6. Use the ***VRaySun*** + ***VRaySky*** system for lighting. If one is not yet practical in its use it is possible to create a lighting setup at will. The exercise's fundamental purpose does not concern lighting techniques.
7. Place ***VRaySun*** by setting parameters as in the image. P.S.: ***VRaySun*** parameters have been refined and modified in the versions following the 1.48. Thus probably these settings might generate a different lighting if one is not using the v1.48.

■ **Figure 3.50**
The new ***VRaySun*** is applied for lighting the scene.

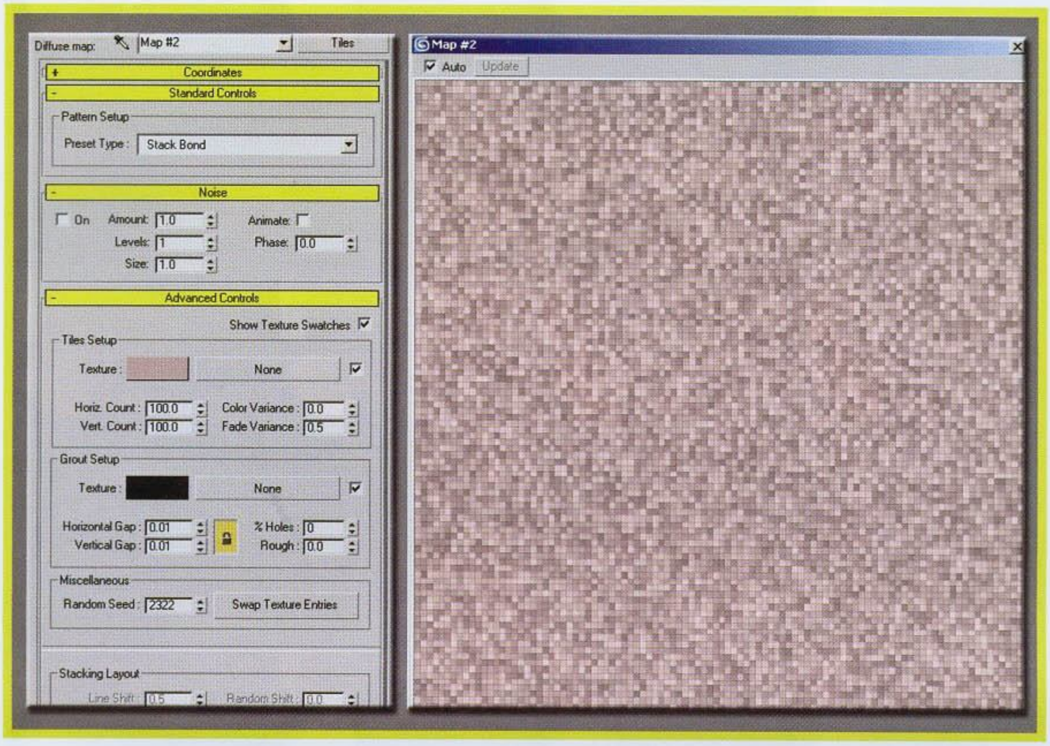


8. A ***skylight*** is also used for lighting the scene. Thanks to the new feature of *VRay* it is possible to use as ***skylight*** the ***VRaySky*** map, which allows one to simulate the sky correctly. Later we will see this object in more detail. Manually moving the position of ***VRaySun*** automatically updates the ***VRaySky*** map, modifying the background so as to coordinate it with the position of the sun. In this case ***VRaySun*** was placed at the sunset height. The ***VRaySky*** was amended to show a blue sky with yellow-orange shades.



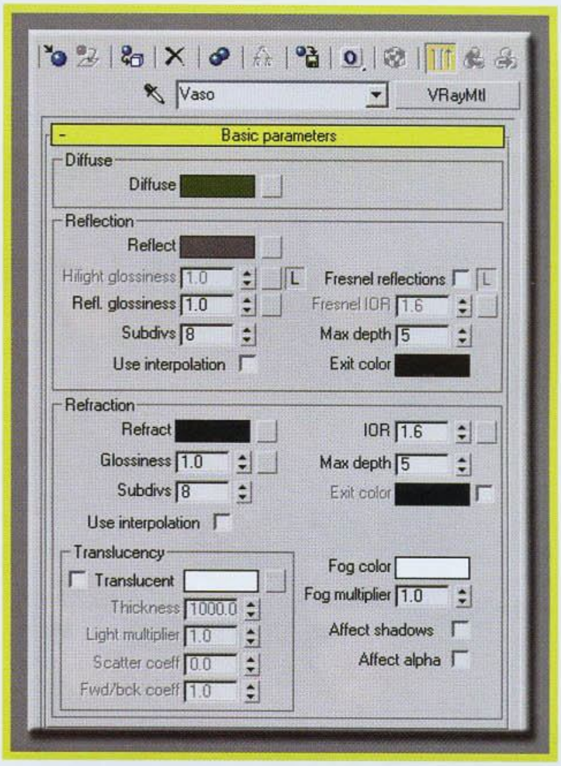
■ **Figure 3.51**
This image shows the ***VRaySky*** shader which is to be used for two purposes: as a map for the background (1) and as a texture to generate GI through ***skylight*** (2). It may have been possible to skip the map in VRay's ***Environment***. In this way the ***skylight***, the reflections and the background would have been handled by 3ds Max's Environment.

9. An UVWMapping modifier, set on Planar with Length =150 and Width =150, is applied on the plane.
10. Apply a ***VRay*** shader with a Tiles map in the ***Diffuse*** channel.



■ **Figure 3.52**
A very dense procedural 3ds Max Tiles map is used as a shader for the support plane. This avoids texture repetition problems, but lays the ***antialiasing*** on the line.

11. The shader of the vases is a dark-green ***VRayMtl*** with a slight reflection.



■ **Figure 3.53**
It is recommended to always apply VRay materials when using VRay as rendering engine.

■ **Figure 3.54**
Test with different antialiasing computing methods.
- - - DVD - - -

■ **Figure 3.55**
Enlarged image.

CONCLUSIONS

Fixed rate	*Subdiv = 1*		*- No Filter*	*........................... 11 min. 32 sec.*	
Fixed rate	*Subdiv = 1*		*- Mitchel filter*	*........................... 13 min. 8 sec.*	
Fixed rate	*Subdiv = 4*		*- Mitchel filter*	*........................... 24 min. 5 sec.*	***(Best quality)***
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 50*	*- No filter*	*... 2 min. 42 sec.*	
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 50*	*- Mitchel*	*... 3 min. 18 sec.*	
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 10*	*- Mitchel*	*... 3 min. 13 sec.*	
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 6*	*- Mitchel*	*... 4 min. 30 sec.*	
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 5*	*- Mitchel*	*... 4 min. 44 sec.*	
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 4*	*- Mitchel*	*... 5 min. 4 sec.*	*(Good quality/time ratio)*
Adaptive QMC	*Min subdiv = 2*	*Max subdiv = 4*	*- Mitchel*	*... 7 min. 20 sec.*	
Adaptive QMC	*Min subdiv = 3*	*Max subdiv = 4*	*- Mitchel*	*.. 13 min. 54 sec.*	*(Better quality/time ratio)*
Adaptive QMC	*Min subdiv =4*	*Max subdiv = 4*	*- Mitchel*	*.. 23 min. 18 sec.*	*(Small difference from prev.)*
Adaptive subdiv	*Min rate = -3*	*Max rate = -3*	*- No filter.........*	*22 sec.*	*(Maximum rendering speed)*
Adaptive subdiv	*Min rate = -3*	*Max rate = -3*	*- Mitchell.........*	*24 sec.*	
Adaptive subdiv	*Min rate = -2*	*Max rate = -3*	*- Mitchell......*	*1 min. 3 sec.*	
Adaptive subdiv	*Min rate = -1*	*Max rate = -3*	*- Mitchell......*	*3 min. 44 sec.*	
Adaptive subdiv	*Min rate = -1*	*Max rate = -2*	*- Mitchell......*	*3 min. 53 sec.*	*(No difference from the prev.)*
Adaptive subdiv	*Min rate = -1*	*Max rate = -1*	*- Mitchell......*	*3 min. 52 sec.*	*(No difference from the prev.)*
Adaptive subdiv	*Min rate = -1*	*Max rate = 0*	*- Mitchell......*	*7 min. 26 sec.*	*(Changes in far away areas)*
Adaptive subdiv	*Min rate = -1*	*Max rate = 1*	*- Mitchell......*	*10 min. 36 sec.*	
Adaptive subdiv	*Min rate = -1*	*Max rate = 2*	*- Mitchell......*	*13 min. 13 sec.*	
Adaptive subdiv	*Min rate = 0*	*Max rate = 2*	*- Mitchell......*	*18 min. 35 sec.*	
Adaptive subdiv	*Min rate = -1*	*Max rate = -1*	*- Mitchell......*	*19 min. 23 sec.*	*(Good quality, but evident problems in areas close to the DOF)*

There are three critical zones in this image:

1. The transition zone between the ***DOF*** of vases in the foreground and the ***Background*** sky.
2. The bodies of two vases in the foreground to which ***DOF*** has been applied with the consequent presence of ***Noise***.
3. The flooring texture.

Fixed rate sampler

By increasing the ***Subdivision*** number, global quality increases too. The noise within the ***DOF*** and ***aliasing*** textures as well drastically decreases. ***Subdivision*** = 4 guarantees an excellent quality.

Adaptive QMC sampler

Render is very fast when setting ***Min subdivs*** = 1 and ***Max subdivs*** = 50. Lowering down to values ***Max subdivs*** = 5 the result does not change that much, and times are almost the same too. Around ***Max subdivs*** = 4 the first relevant changes appear: noise starts to reduce. But for obtaining satisfying results, ***Min subdivs*** = 2 must be set again. In this way the noise in the ***DOF*** on the vase is considerably reduced, and it almost vanishes for ***Min subdivs*** = 4.

Adaptive subdivision sampler

With default values ***Min rate*** = -3 and ***Max rate*** = -3, times are extremely low, but quality as well. Progressively increasing the ***Min rate***, up to ***Min rate*** = -1, fairly evident improvements are visible, but rendering time drastically increases too.

Increasing ***Max subdivs*** up to ***Max subdivs*** = -1 does not make any difference. Just by setting its value to ***Max subdivs*** = 0 it is visible that "far away" pixels start to be more defined. A quality comparable to the one obtained with the sampling ***Adaptive QMC Min rate*** = 4 ***Max rate*** = 4 might be achieved in 20 min. with ***Min subdivs*** = 0 and ***Max subdivs*** = 3. In any case, the zone between the ***DOF*** and the background is low quality. In fact a sort of line, edge, marking the boundary of the ***DOF*** can be noticed. This effect does not occur with the two previous methods. Concluding, the method ***Adaptive subdivision sampler*** tends to mix, spread the noise, smoothing it out. But sometimes, with this method, "stains" appear, in addition to extremely long time being required. As already widely discussed, this kind of sampling is to be preferred for low detail scenes.

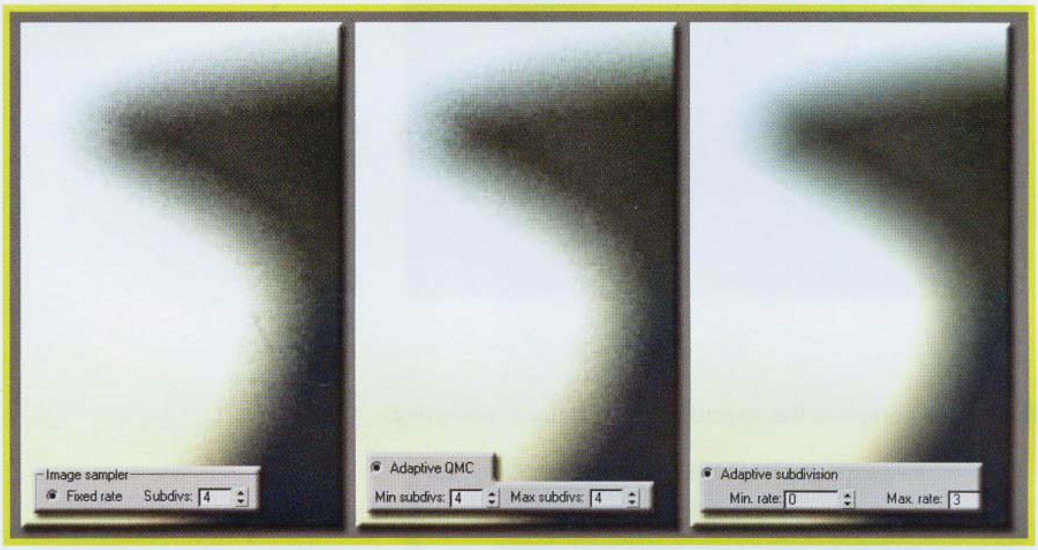


Figure 3.56
It can be noticed how the smooth nature of the ***Adaptive subdivision*** creates a sort of boundary between the ***DOF*** and the ***Background***. With other methods the ***DOF*** tones down more delicately.

Example 6: scene without textures and di DOF

The next scene shows the effectiveness of using the ***Adaptive subdivision sampler*** method.

4. In the frontal view create a Sphere object with radius of 40 units.

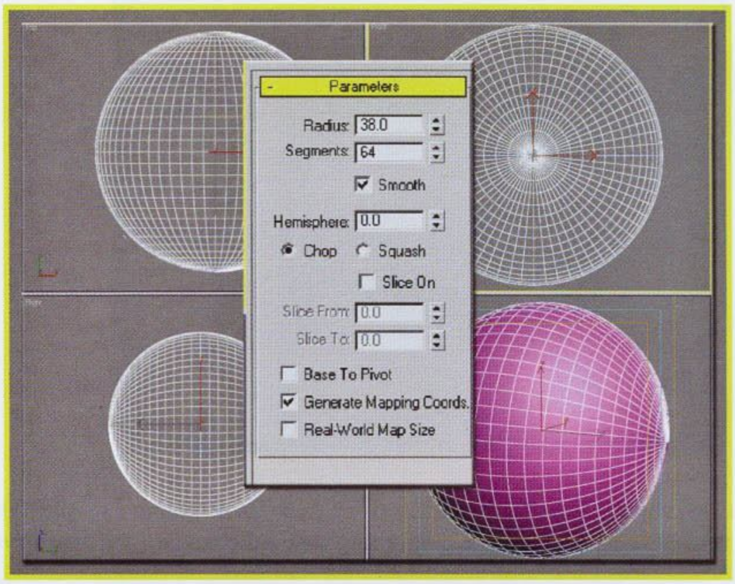
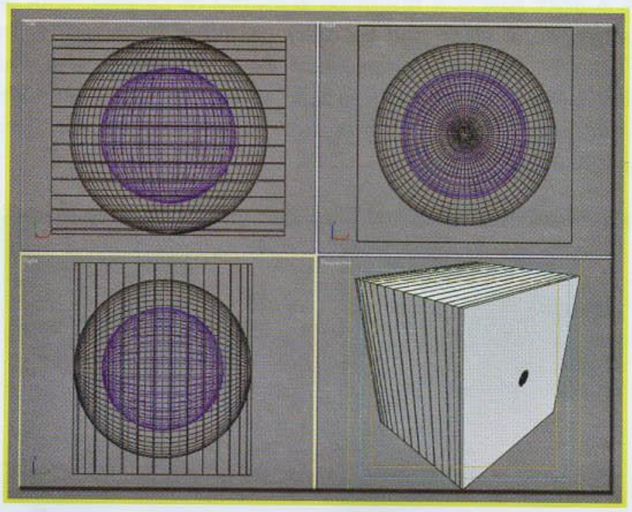


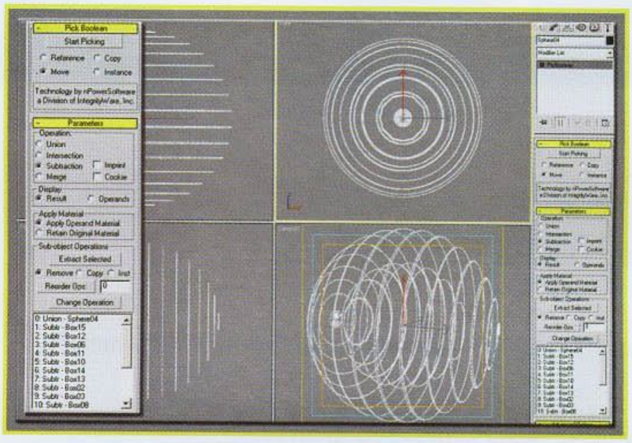
Figure 3.57
Parameters for the creation of the sphere.

5. Create 15 boxes which will be used as volumes to be subtracted to the sphere which has just been created. Starting from the outermost ones, the thickness is, respectively, 1.7-2.9-5-6-6.5-6.7-6.7.



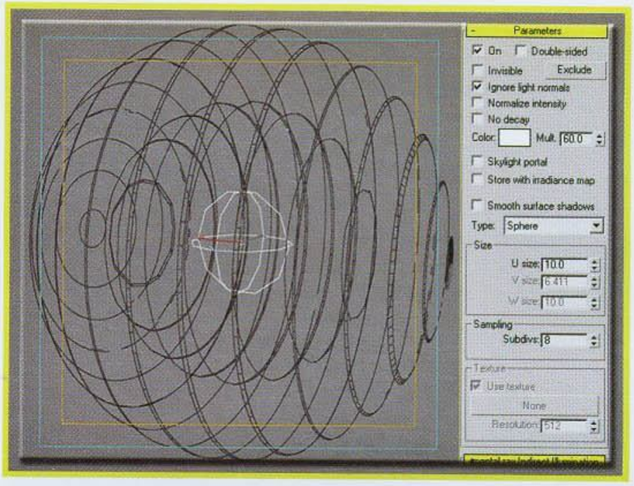
■ **Figure 3.58**
15 Boxes used as objects for the Boolean subtraction. Inside the sphere another sphere of 28 units has been created. It will be subtracted to create an internal space, inside which the luminous source is to be positioned.

6. Use the tool ProBoolean to complete the operation of the lamp's modelling.



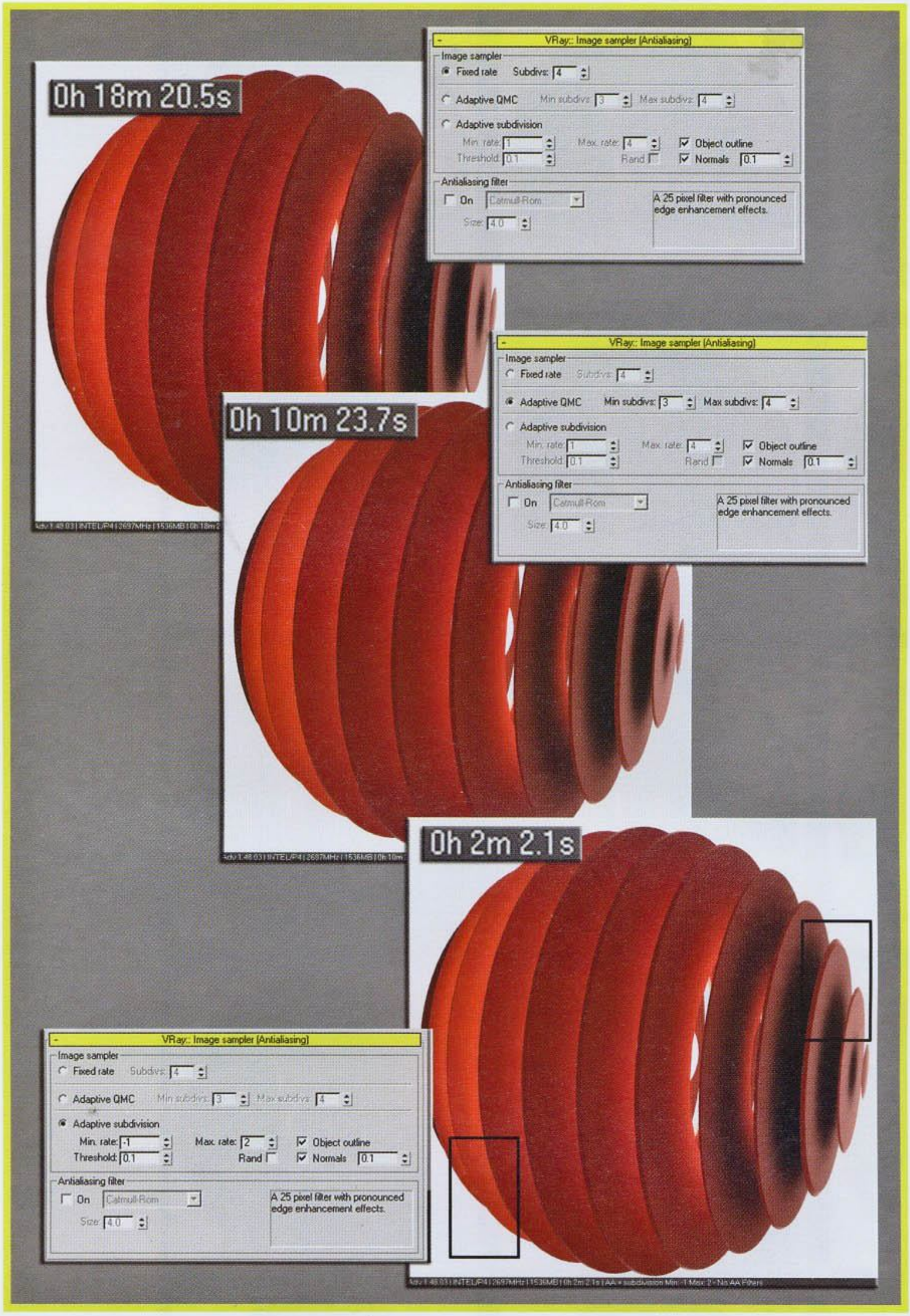
■ **Figure 3.59**
ProBoolean object.

7. Use the spherical ***VRayLight*** as luminous source, positioning it at the centre of the model.



■ **Figure 3.60**
Setting of the ***VRayLight***. Throughout the book, each single parameter contained in it will be explained.

8. The material is a common ***VRayMtl*** with a red light ***Diffuse*** color. For the Global Illumination only the ***skylight*** of ***VRay*** is to be activated, with a computing method based on the ***Irradiance Map*** and the ***QMC*** with default values.



■ **Figure 3.61**
Summarizing image: the best method among the three ones we have seen is the ***Adaptive subdivision***. In 2 min. only it is able to produce a fairly good ***antialiasing***. The only fault is the slightly "stained" surface near the luminous source, near the ***Area Shadows.*** As we have said more than once, when blurry effects are present, the best method is certainly not the ***Adaptive subdivision***. But in this scene most of the surfaces are smooth and therefore the employment of this sampling method is the best solution.
In the two dark squares the areas of interest are marked.

■ **Figure 3.62**
Test on ***antialiasing*** using the three sampling methods.

■ **Figure 3.63**
Image of uniting process with enlargement.

CONCLUSIONS

Fixed rate	*Subdiv = 1 -*	*No filter*	....	*1 min. 58 sec.*	
Fixed rate	*Subdiv = 2 -*	*No filter*	....	*6 min. 10 sec.*	
Fixed rate	*Subdiv = 4 -*	*No filter*	....	*18 min. 20 sec.*	
Fixed rate	*Subdiv = 4 -*	*Catmull*	....	*21 min. 44 sec.*	*(Good quality)*

Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 6 - No filter...*	*1 min.*	*17 sec.*	*(Very low quality)*
Adaptive QMC	*Min subdiv = 1*	*Max subdiv = 4 - No filter ...*	*1 min.*	*16 sec.*	
Adaptive QMC	*Min subdiv = 2*	*Max subdiv = 4 - No filter ...*	*4 min.*	*40 sec.*	*(Very low quality of edges and shadows)*
Adaptive QMC	*Min subdiv =3*	*Max subdiv = 4 - No filter ...*	*10 min.*	*23 sec.*	*(Good quality)*
Adaptive QMC	*Min subdiv =3*	*Max subdiv = 4 - Catmull ..*	*15 min.*	*22 sec.*	*(Best quality)*

Adaptive subdiv	*Min rate = -3*	*Max rate = -3 - No filter.........*		*10 sec.*	*(Very low quality – Very short time)*
Adaptive subdiv	*Min rate = -3*	*Max rate = 2 - No filter .....*	*2 min.*	*28 sec.*	*(Very good quality)*
Adaptive subdiv	*Min rate = -1*	*Max rate = 2 - No filter .....*	*2 min.*	*1 sec.*	*(Better quality/time ratio)*
Adaptive subdiv	*Min rate = 1*	*Max rate = 2 - No filter .....*	*8 min.*	*42 sec.*	
Adaptive subdiv	*Min rate = 1*	*Max rate = 4 - No filter ...*	*11 min.*	*7 sec.*	
Adaptive subdiv	*Min rate = 1*	*Max rate = 4 - Catmull ...*	*15 min.*	*26 sec.*	*(Highter quality)*

There are two critical zones in this image:

9. The area lit by the ***VRayLight***, generating an ***Area Shadows*** in the zones nearest to the luminous source.
10. The outermost thin layers, flat and not detailed at all.

Fixed rate sampler

In this scene the method ***Fixed rate*** rewards the uniformity of geometries and materials. In very poorly detailed areas, the sampler takes long time for computing ***antialiasing***. In 21 min. it generates a correct image.

Adaptive QMC sampler

As for previous examples, in this case too the sampler, very similar to the ***Fixed rate sampler***, is not at the top of its possibilities. It persists in working on areas that do not need particular care. In 10 min. only a satisfying image is obtained.

Adaptive subdivision sampler

Speed in easy zones is its strong point. Values of ***Min subdivs*** = -1 and ***Max subdivs*** = 2 are enough for generation of a rendering ready to be used. Higher values make the image more accurate in areas naturally more in favour of the ***Adaptive QMC***, as happens near the ***Area Shadows***. In these zones a slight stained noise forms with values ***Min subdivs*** = -1. In any case it is not a problem, because of its little extension.

Example 7: textures and antialiasing

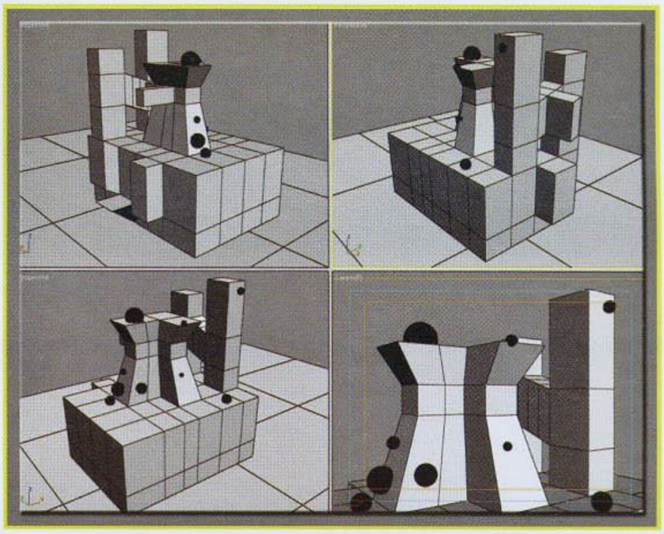
With the next test we want to illustrate the effects of the parameter ***Color threshold***, using as a sampling method the ***Adaptive subdivision***. By nature, *VRay* does not support the supersampling options implemented in 3ds Max for texture filtering. Instead, it uses the three proprietary methods seen before and the relative options.

By default *VRay* applies ***antialiasing*** to the whole image, including textures, reflections, etc... The parameter ***Color threshold*** controls the quality of ***antialiasing*** specifically for textures, procedural or not, if the ***Adaptive subdivision*** method is used.

When using high levels of the ***Color threshold*** parameter, *VRay* does not apply ***antialiasing*** to textures. This may be useful when quick renderings must be carried out. It must be reminded that this disables ***antialiasing*** for ***VRayShadows***, reflections and refractions etc… as well.

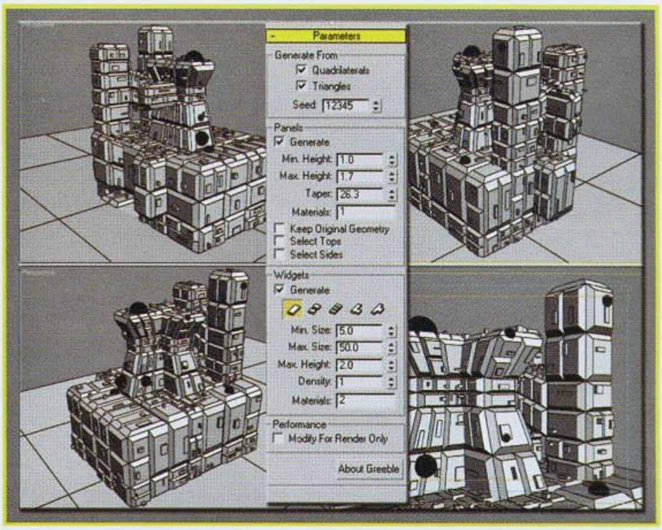
In the following examples, ***Adaptive subdivision*** is used, with ***Min rate*** = -3 and ***Max rate*** = 2. With this procedure a high value of ***undersampling*** is obtained, and this allows one to better observe the effects of the variation of the parameter ***Color threshold***.

1. The scene is composed by two main geometric objects. The surface plane and the polygonal building. Ten spheres have been added to enrich the scene. For those who are not yet experts, some passages might be obscure. There is no problem. The only purpose of this exercise is to understand the parameter ***Color threshold***. Everything written from now on is aimed at outlining how to create and set a scene. In the next chapters everything still unclear will be explained in-depth.



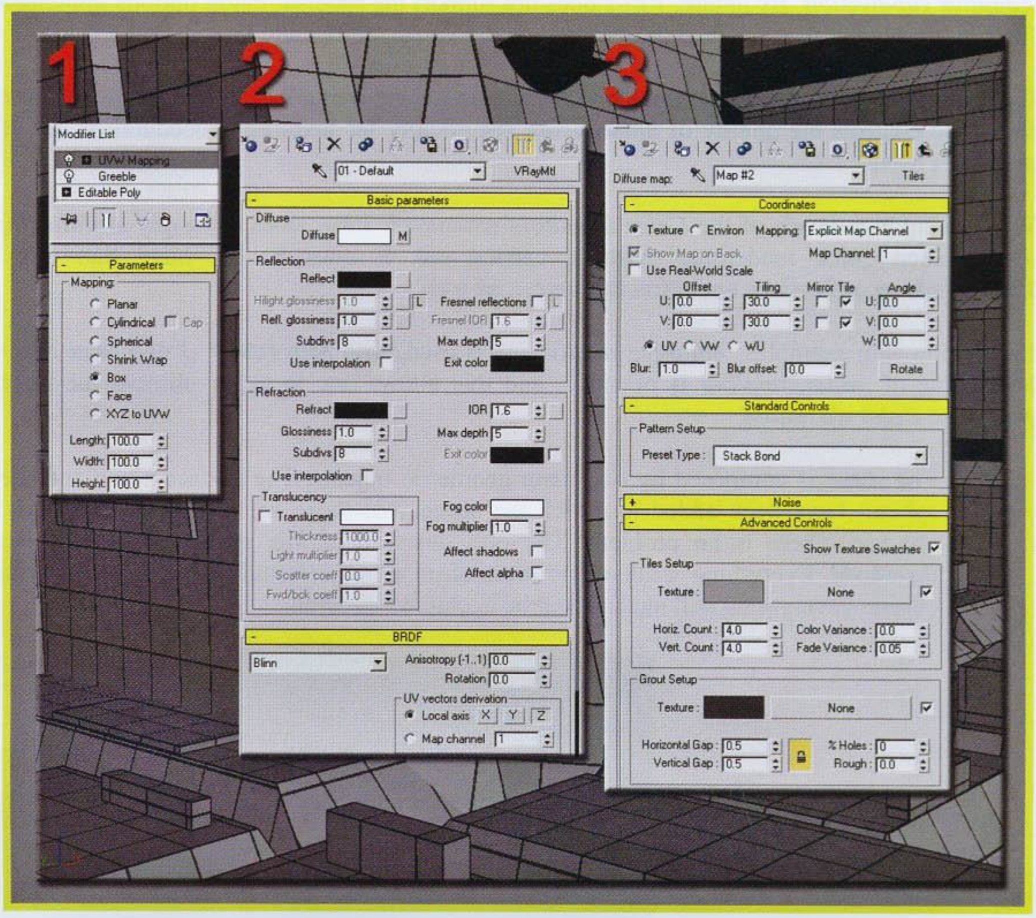
■ Figure 3.64
The scene has been created starting from a box of a size of 50x70x30 units. Subdivided in several parts, some polygons have been extruded more than once to create towers.

2. Using the free Greeble plug-in, some details have been added to enrich the main structure.



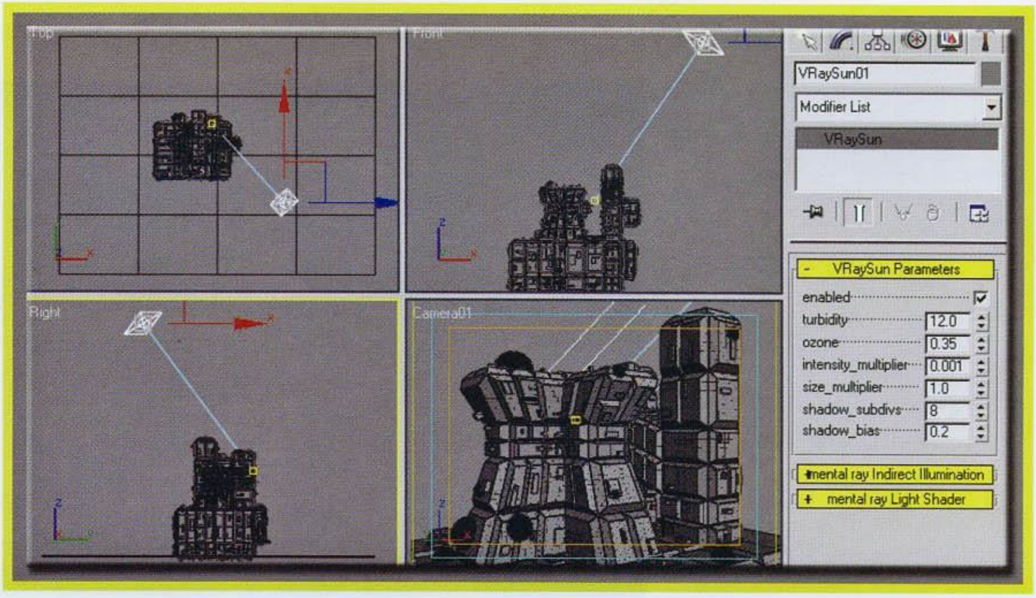
■ Figure 3.65
The plug-in Greeble was born from the genius of the same developer of 3D Studio, Tom Hudson, many 3ds Max releases ago. With this plug-in procedural details on mesh object surfaces can be generated.

3. After, a UVW Mapping modifier (1) was added, and a ***VRayMtl*** shader (2) was created, in the ***Diffuse*** channel where a Tiles map (3) has been inserted.



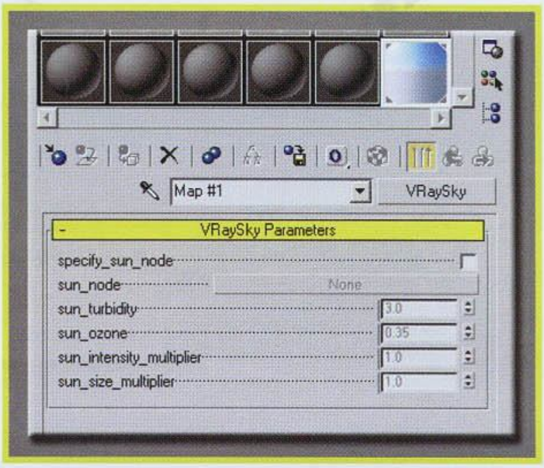
■ **Figure 3.66**
Set the parameters as in the image. Further on in the book each single parameter of ***VRay Material*** will be explained.

4. Once again, for lighting, the splendid system of solar simulation introduced with the ***v1.48.xx*** of ***VRay*** has been used.



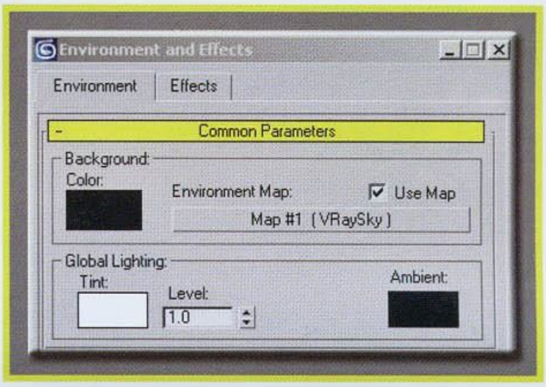
■ **Figure 3.67**
The position of ***VRaySun***.

5. For the generation of the GI, the ***VRaySky*** map is to be used.



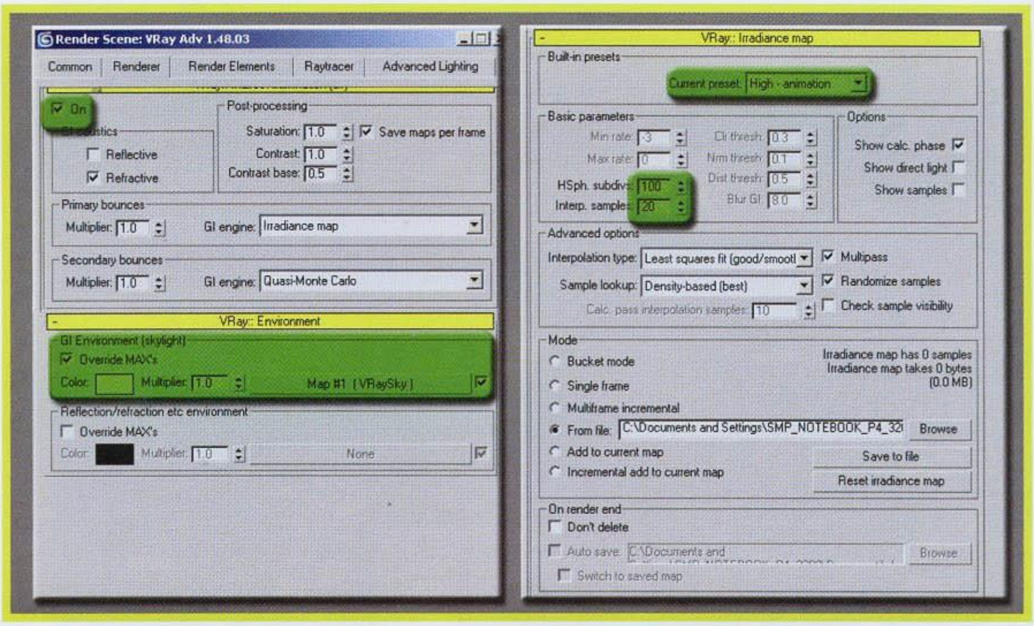
■ **Figure 3.68**
The map ***VRaySky*** s very simple and has few parameters. They are deactivated by default.

6. Copy the ***VRaySky*** in the slot Environment of 3ds Max. This allows the background of the rendering to be generated by the ***VRaySky*** map.



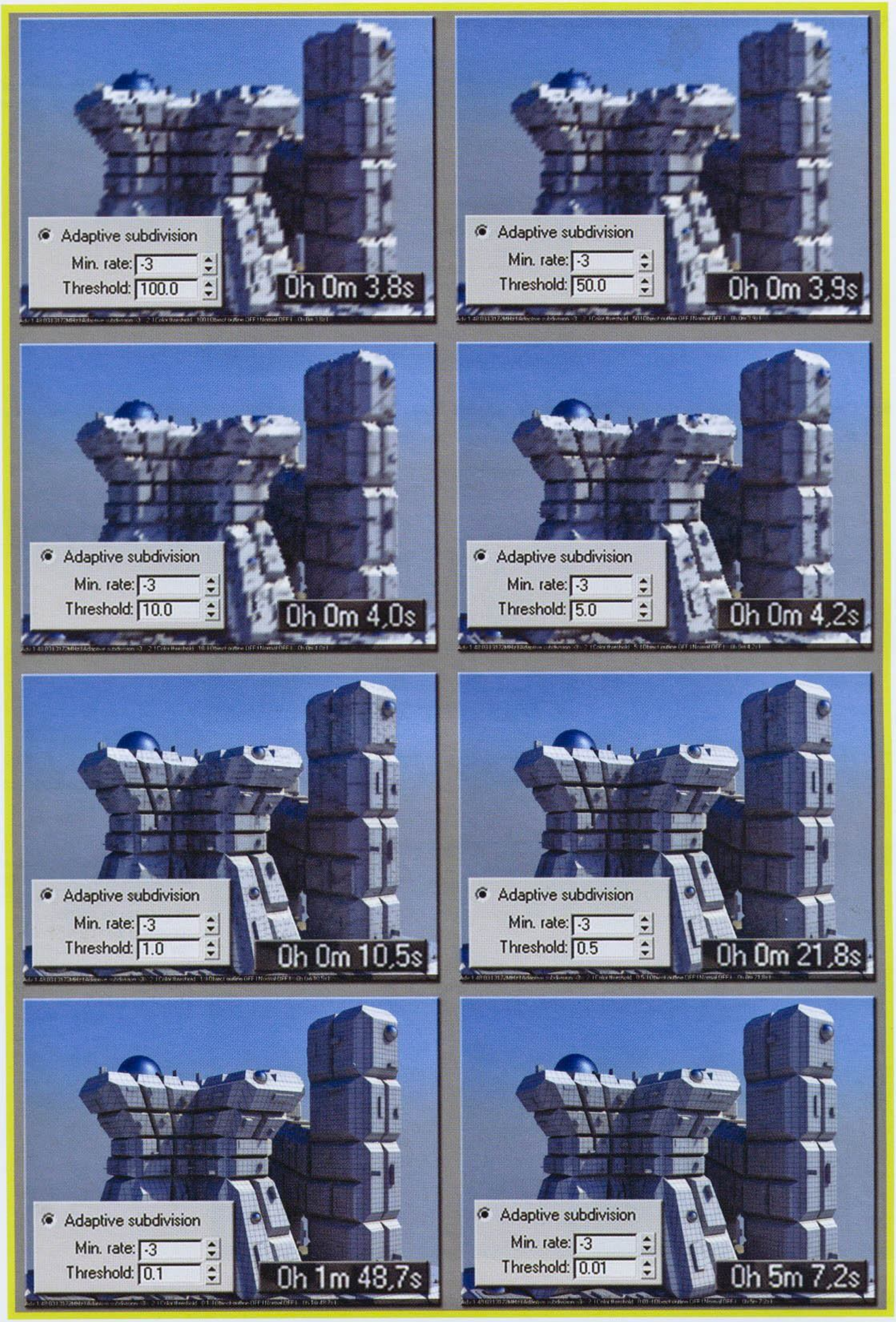
■ **Figure 3.69**
Copying the map on request guarantees that the change of the ***VRaySky*** map is updated in the Environment of 3ds Max as well.

7. The ***VRaySky*** is copied in the slot ***Environment*** of *VRay*. This way ***skylight*** is generated by the ***VRaySky*** map. This last passage could have been ignored. We will understand why later on in the book, in the chapter devoted to the ***Environment***.

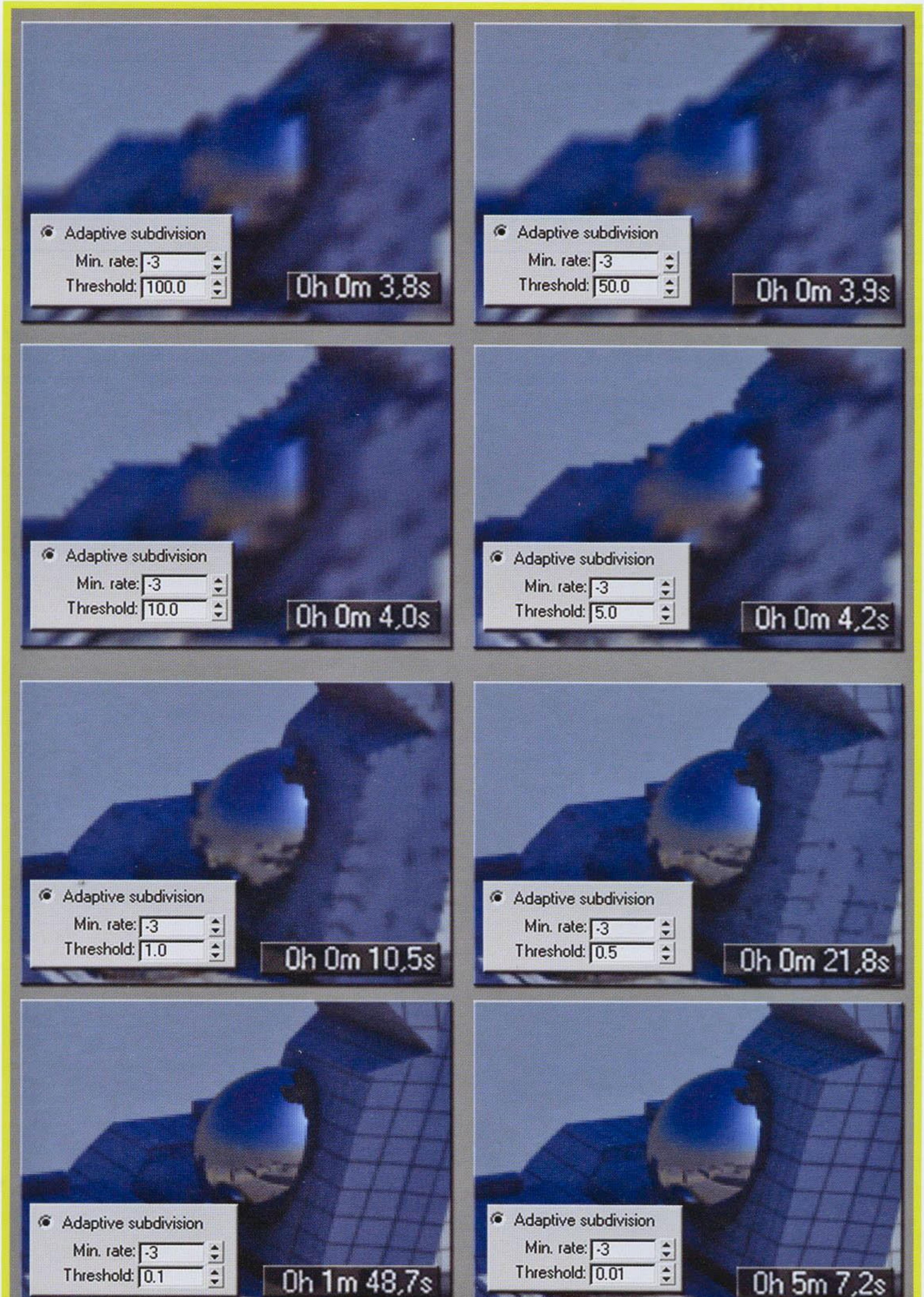


■ **Figure 3.70**
The purpose of this exercise is not to explain in detail each parameter in this image. Certainly they will be clearer once the chapter concerning the ***Irradiance Map*** has been covered. It is advisable to go on and later go back to what was not clear once the necessary knowledge has been gained.

8. Now the scene is ready to be rendered.



■ **Figure 3.71**
Scene rendered with different values of the ***Threshold*** parameter. ***Adaptive Subdivision*** is set to ***Min rate*** = -3 and ***Max rate*** = 2. The ***Object outline*** option is deactivated.



■ **Figure 3.72**
Union image with enlargement.

CONCLUSIONS

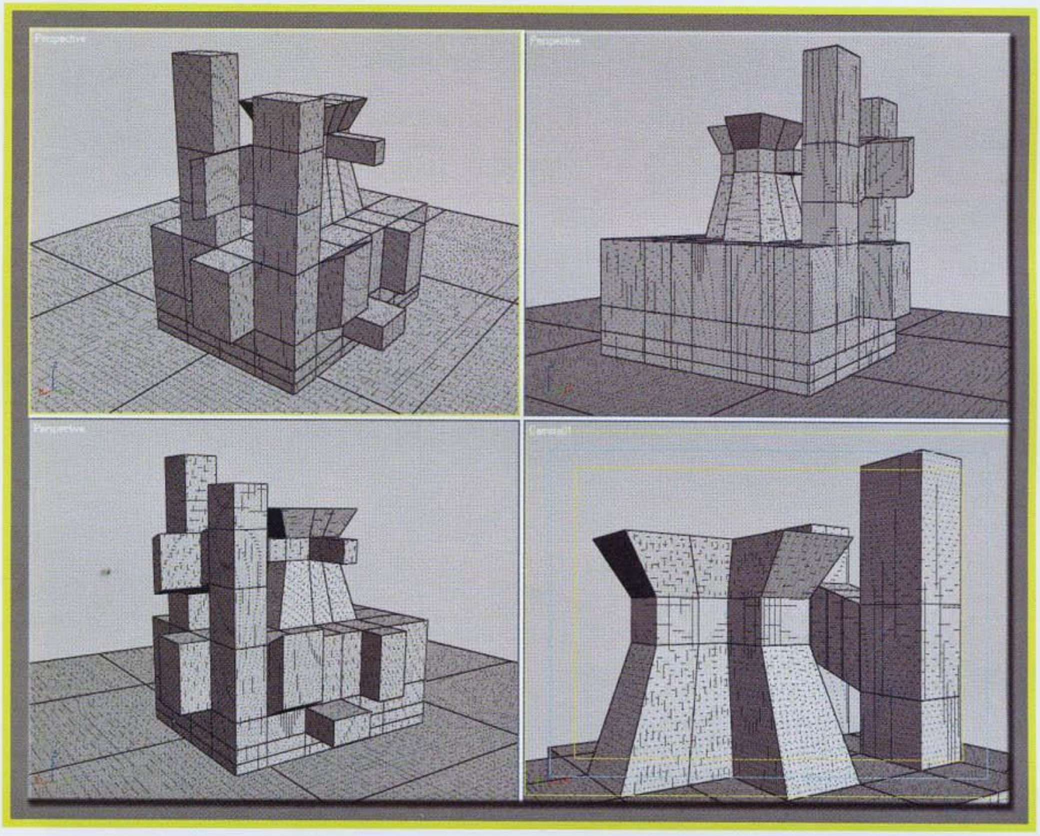
Threshold = 100	*...*	*04 sec.*	
Threshold = 50	*...*	*04 sec.*	
Threshold = 10	*...*	*04 sec.*	
Threshold = 5	*...*	*04 sec.*	
Threshold = 1	*...*	*10 sec.*	
Threshold = 0.1	*...*	*01 min. 48 sec.*	*(Better quality/time ratio)*
Threshold = 0.01	*...*	*05 min. 07 sec.*	*(Highter quality)*

In this image the most difficult areas to be rendered are those in shadow. With values higher than 0.1 a complete loss of information occurs, thus giving a low-quality rendering. With value set to 0.1, instead, most of the interspaces between tiles are correctly perceived, even though this does not happen in critical areas in shadow. To obtain the highest quality it is necessary to lower the values even more. ***Threshold*** = 0.05 could be the right setting. ***Threshold*** = 0.01 guarantees the maximum quality, but at the expense of high times.

Example 8: Object outline and Normal

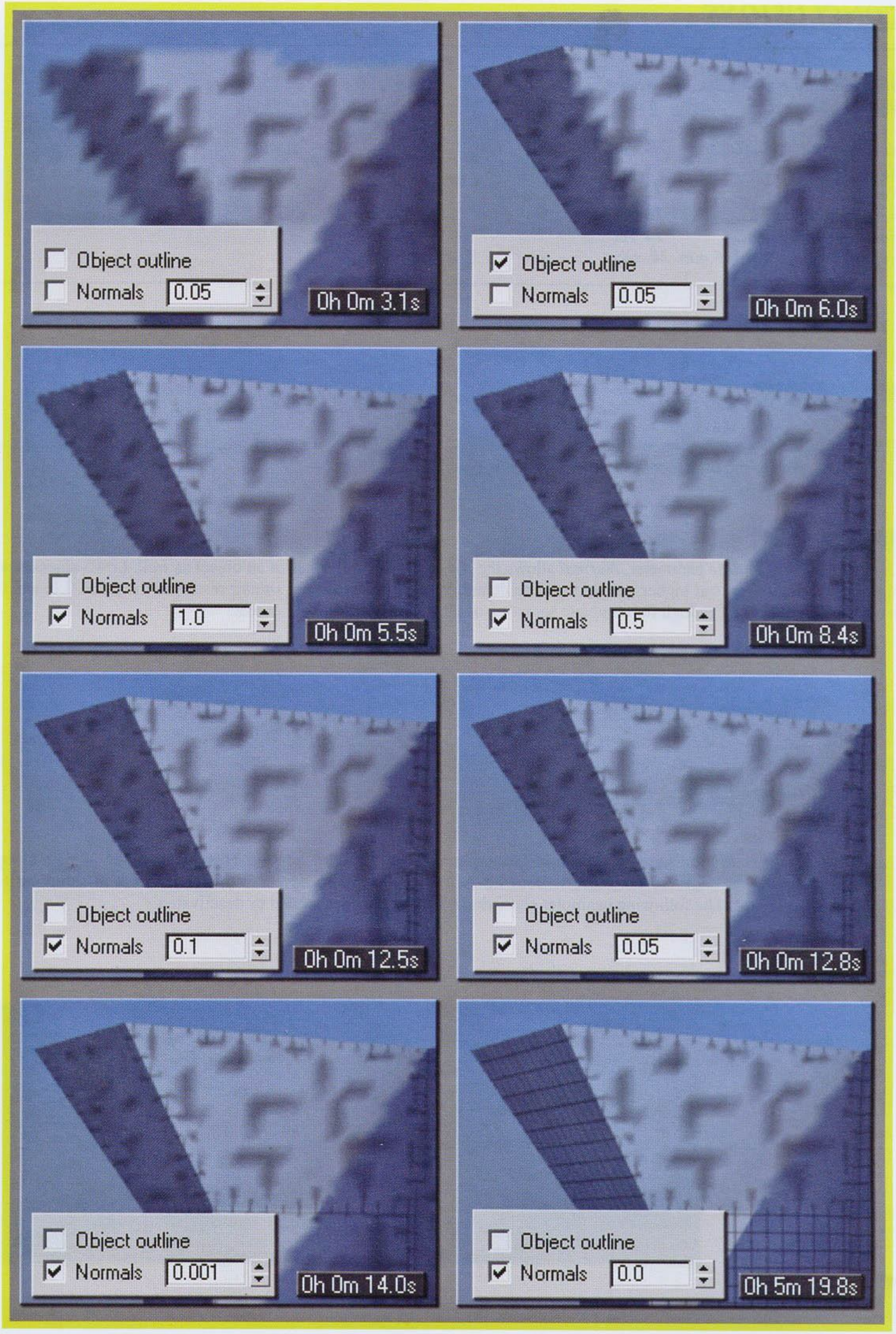
In the previous example, the only parameter handling the quality of ***antialiasing***, apart from ***Min rate*** and ***Max rate***, was ***Color threshold***. This option controls the quality of ***antialiasing*** for the whole image, including mesh edges. High values meant poor-quality ***AA***. This was so because the parameter ***Object outline*** was disabled. Activating it, even with high values of ***Color threshold***, in any case it will be possible to obtain excellent quality of the ***AA*** along the edges of the object, while the ***AA*** of the internal areas of the objects will always be managed by ***Color threshold***.
In the case of small-size objects, the activation of this parameter might not help rendering times. In these cases it is advisable to deactivate it, leaving the task of the ***antialiasing*** to the ***Color threshold***. On the contrary, in the presence of many very detailed textures, the activation of this parameter combined with high values of ***Color threshold*** might be very useful, because it would leave the edges sampled with the ***AA***, and the internal areas without it. In case of a detailed texture the difference would be negligible; moreover rendering times would decrease considerably.
The ***Normal*** parameter allows to choose further areas where to apply the ***AA***; that is, it manages the ***antialiasing*** of the internal edges of the geometries, in order with the angle between the polygons composing them.

The examined scene is the same of the previous one, except for the modifier ***Greeble***, which is deactivated now. This way it is easier to point out the corners of the mesh. At this point the main edges, internal and external, of the two buildings, appear more visible.



■ **Figure 3.73**
For illustrating the parameters ***Object outline*** and ***Normal***, the previous scene was used, except for the modifier ***Greeble***, which is deactivated.

■ **Figure 3.74**
Scene rendered with different values of the ***Normals*** parameter. ***Min rate*** = -3, ***Max rate*** = -2 and ***Threshold***= 50.



■ **Figure 3.75**
Union image with enlargement.

CONCLUSIONS

Object outline = OFF	.....	*3 sec.*
Object outline = ON	.....	*6 sec.*
Normal = 1	......	*5 sec.*
Normal = 0.5	......	*8 sec.*
Normal = 0.1	......	*12 sec.*
Normal = 0.05	......	*13 sec.*
Normal = 0.001	......	*14 sec.*
Normal = 0	......	*5 min. 20 sec.*

OBJECT OUTLINE

The activation or deactivation of this parameter allows the ***AA*** of the external edges of an object only. As a consequence, rendering time increases: in this case it is doubled.

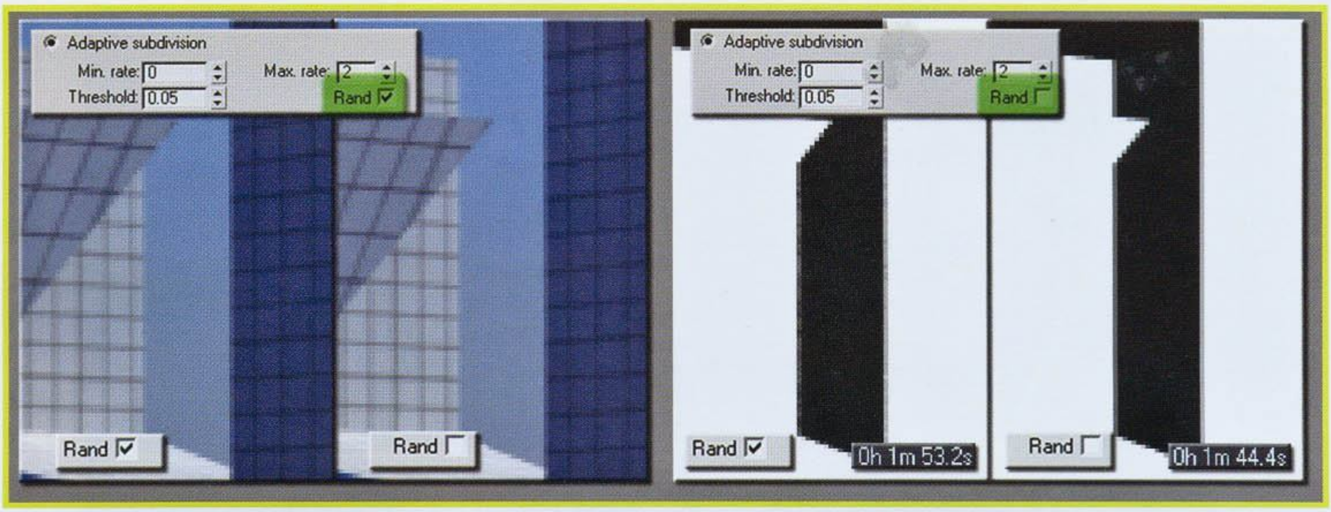
NORMAL

Unlike the ***Object outline*** parameter, ***Normal*** allows the ***AA*** of the internal edges of an object as well. Low values mean higher computing time, but higher quality too. The default value 0.05 gives very similar results to the value 0.1, which proves in any case to be enough for complete sampling of all internal edges of the object.

Example 9: random Antialiasing

Usually *VRay* distributes the samples for ***antialiasing*** in a very rigid and linear way. This phenomenon is plain along vertical and horizontal edges of the objects. Sometimes it is preferable to have certain randomness in sampling to achieve a better image. The following examples show the parameter ***Rand*** activated or deactivated.

■ **Figure 3.76**
Scene rendered first with the option ***Rand*** active, and then deactivated.



■ **Figure 3.77**
Examining the results through the ***alpha*** channel will help grasp a better understanding of this subject.

CONCLUSIONS

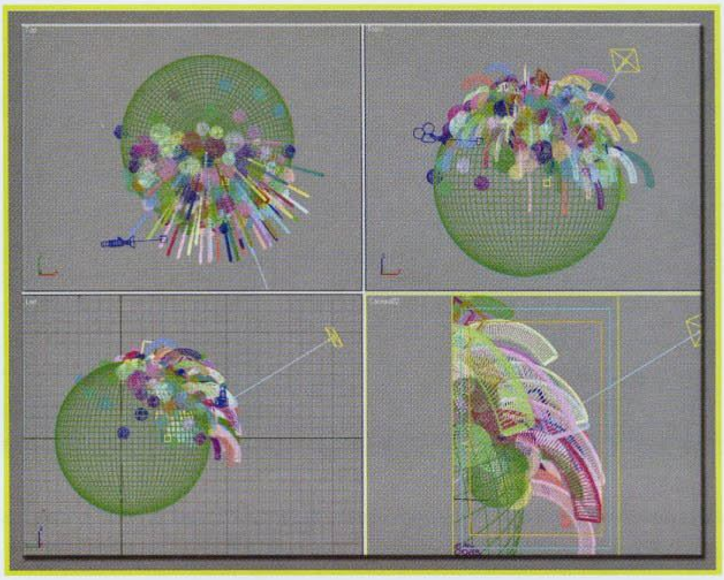
As can be seen through the channel ***alpha***, the use of the ***Rand*** function has generated a random ***antialiasing*** along vertical edges of the building. By deactivating the parameter ***Rand*** rendering time decreases.

Example 10: filters

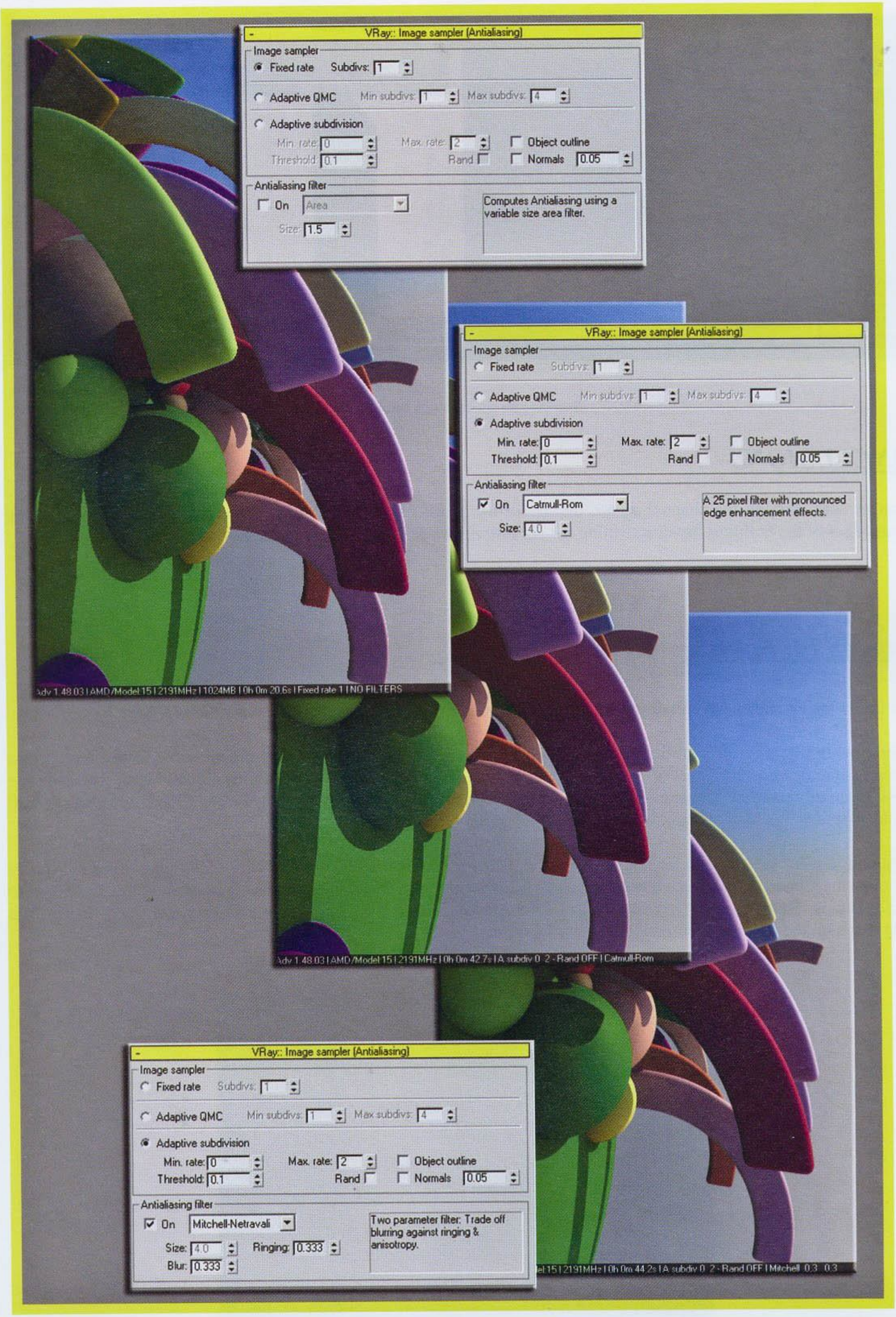
In the following examples we will describe the main differences between the 11 filters available in *VRay*.

Rendering with filters is different from rendering without filters and then applying post-production effects after, such as Sharpen or Photoshop. The filters of 3ds Max are applied at ***subpixel*** level. Thus, applying filters during the rendering process, definitely makes the image more accurate than what it would be using post-production filters. *VRay* can use all filters available in 3ds Max, except for the ***Plate match*** filter, which is not supported.

A very simple scene is used, and for this reason the explanations about its making will not be explored. It is a big sphere to which some Chamfer Boxes have been aligned randomly along normals to the faces. A Bend modifier is applied to Chamfer Boxes. Default colors have been kept for materials, while the illumination has been achieved thanks to a system based on the ***VRaySky*** method (see previous scenes).



■ **Figure 3.78**
3ds Max's viewport image.



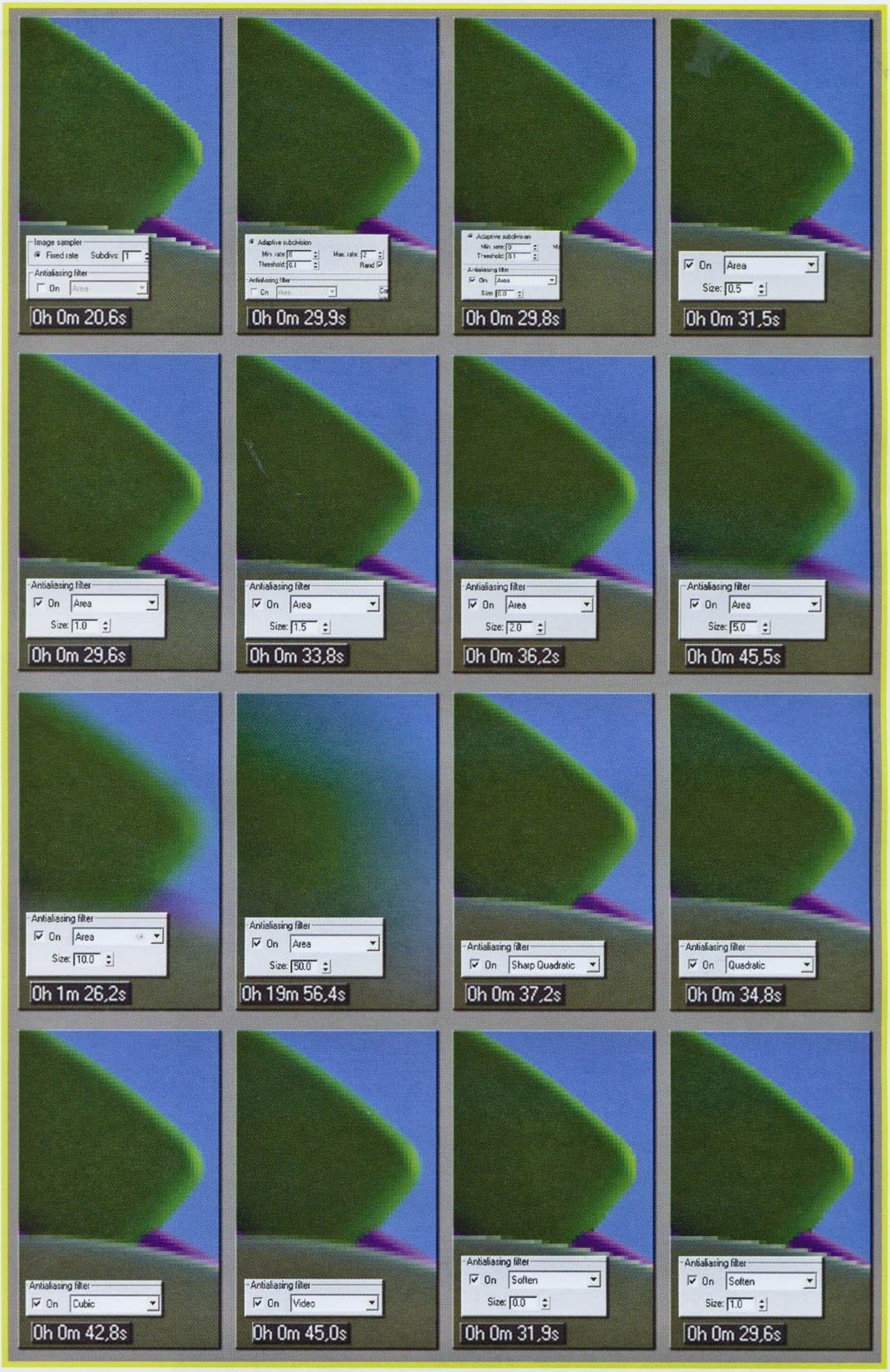
■ **Figure 3.79**
The first image has been rendered without ***antialiasing***. In the two remaining ones the filters usually used for static renders have been used.

■ **Figure 3.80**

This image summarizes the filters available in 3ds Max in 5 images.

The purpose is to compare the numerous combinations.

--- DVD ---



■ **Figure 3.81**
Detail with scale factor 10X.

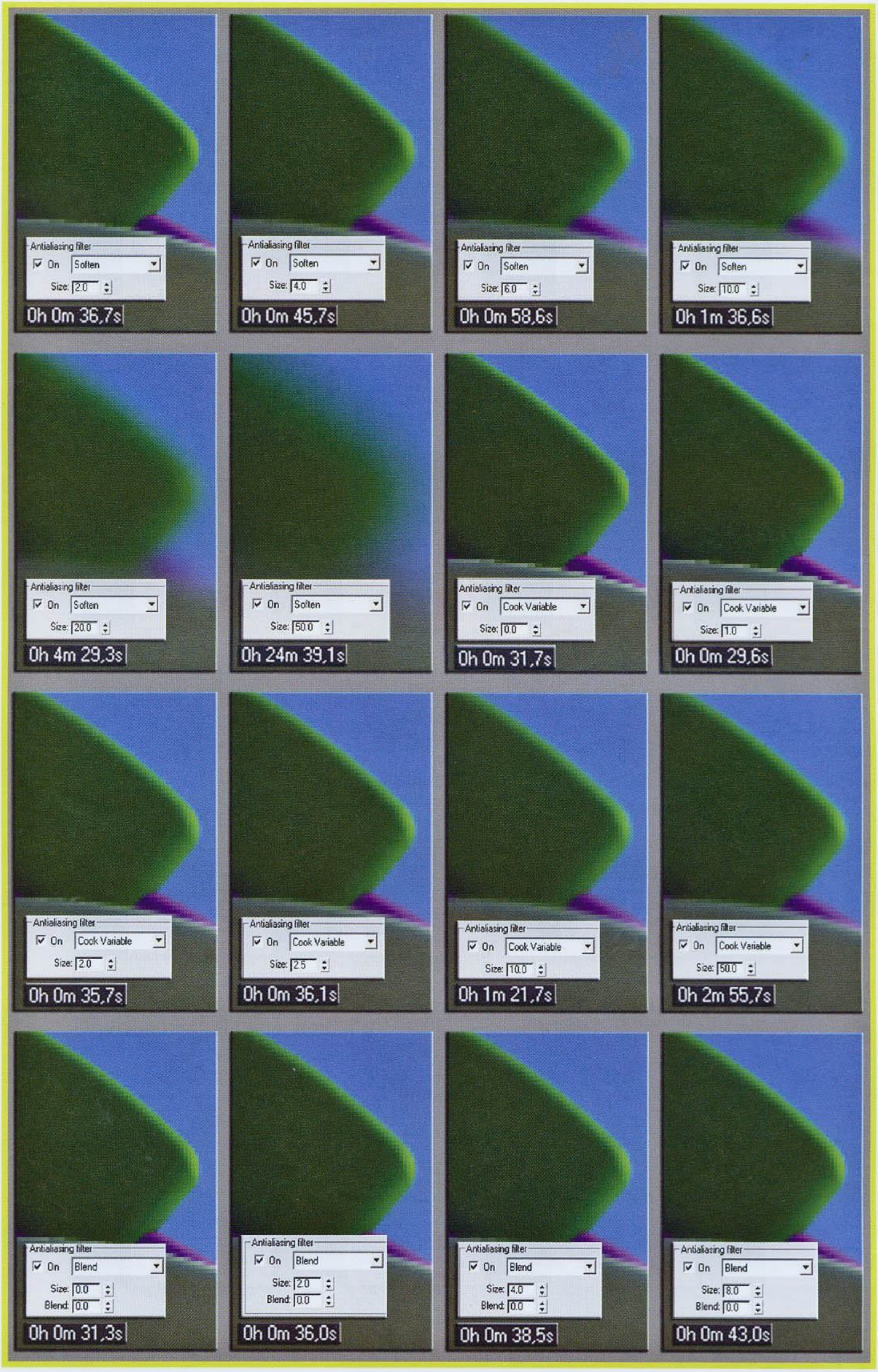
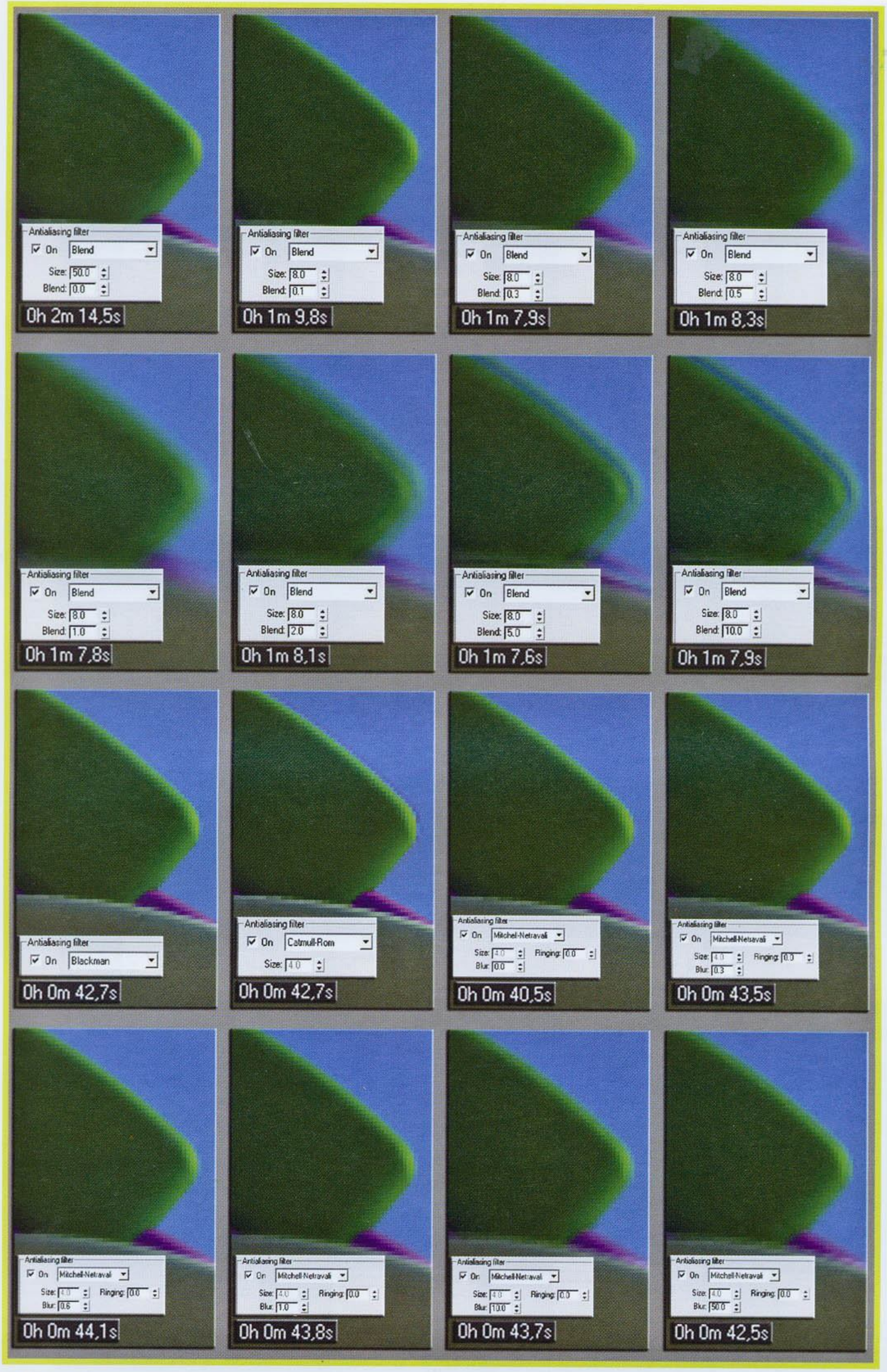
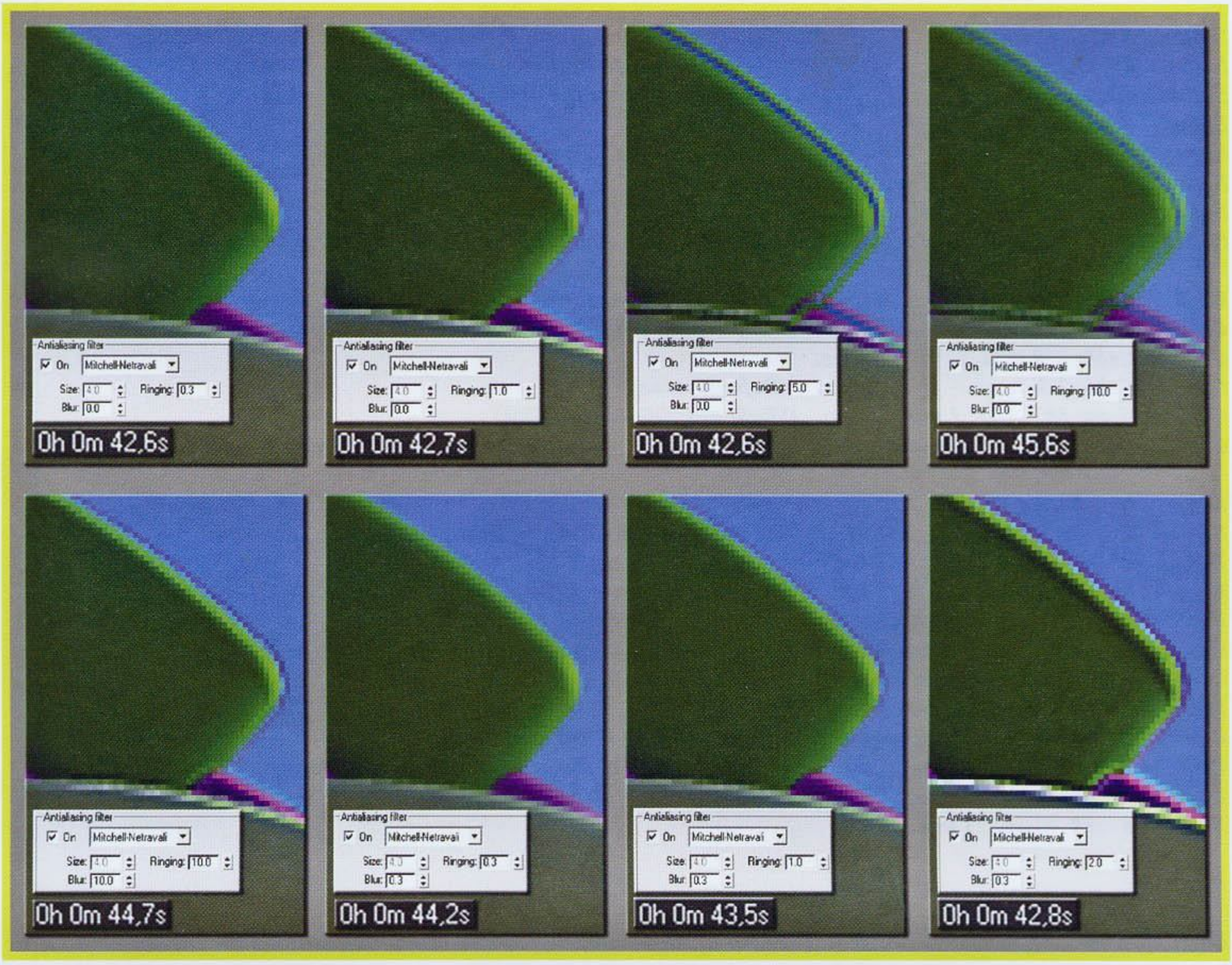


Figure 3.82
Detail with scale factor 10X.



■ **Figure 3.83**
Detail with scale factor 10X.



■ **Figure 3.84**
Detail with scale factor 10X.

CONCLUSIONS

Fixed rate - Filters OFF .... 20 sec. (No filter applied)

Adaptive subd - Filter OFF .... 29 sec. (Application of the AA without any filter)

Area 0 .... 29 sec. (No variation compared to the previous one)
Area 0.5 .... 31 sec.
Area 1 .... 30 sec.
Area 1.5 .... 33 sec. (Default value - Softer edges)
Area 2 .... 36 sec.
Area 5 .... 45 sec. (Edges start to be unfocused)
Area 10 .... 1 min. 26 sec. (Visible blurring of the edges)
Area 50 .... 19 min. 56 se.c (Complete blurring of edges and of the same image)

Sharp Quadratic .... 37 sec. (This and the three following filters lead to very similar results)

Quadratic .... 34 sec.

Cubic .... 42 sec.

Video .... 45 sec.

Soften 0	*31 sec.*	*(Very similar to Area = 0, but with slightly "harder" edges)*
Soften 1	*29 sec.*	*(Very similar to Area = 1, but with slightly "harder" edges)*
Soften 2	*36 sec.*	*(Value similar to the default one for Area = 1.5)*
Soften 4	*45 sec.*	
Soften 6	*58 sec.*	*(Default value - Fading edges)*
Soften 10	*1 min. 36 sec.*	
Soften 20	*4 min. 29 sec.*	
Soften 50	*24 min. 39 sec.*	*(Very smooth edges, but not as much as in the case of Area = 50)*
Cook Variable 0	*31 sec.*	*(Very similar to Area = 0.5)*
Cook Variable 1	*29 sec.*	*(No substantial difference compared to the previous one)*
Cook Variable 2	*35 sec.*	
Cook Variable 2.5	*36 sec.*	*(Default value)*
Cook Variable 10	*1 min. 21 sec.*	*(Slight blurring of the edges)*
Cook Variable 50	*2 min. 55 sec.*	*(No substantial difference compared to the previous one)*
Blend 0 0	*31 sec.*	*(Slight AA of the edges)*
Blend 2 0	*36 sec.*	*(Size > 2 does not substantially affect the AA)*
Blend 4 0	*38 sec.*	
Blend 8 0	*43 sec.*	
Blend 50 0	*2 min. 14 sec.*	
Blend 8 0.1	*1 min. 9 sec.*	*(Edges start to be very softly unfocused)*
Blend 8 0.3	*1 min. 8 sec.*	*(Default values)*
Blend 8 0.5	*1 min. 8 sec.*	
Blend 8 1	*1 min. 7 sec.*	*(Limit of splitting of edges)*
Blend 8 2	*1 min. 8 sec.*	*(An edge parallel to the original one starts to form)*
Blend 8 5	*1 min. 7 sec.*	
Blend 8 10	*1 min. 7 sec.*	
Blackman	*42 sec.*	
Catmull-Rom	*42 sec.*	*(Very hard edges – formation of a bright halo around the edges)*
Mitchell 0 0	*40 sec.*	
Mitchell 0.3 0	*43 sec.*	
Mitchell 0.6 0	*44 sec.*	
Mitchell 1 0	*43 sec.*	*(Blurring of edges are visible)*
Mitchell 10 0	*43 sec.*	*(Very visible blurring)*
Mitchell 50 0	*42 sec.*	*(No difference from the previous one)*
Mitchell 0 0.3	*42 sec.*	
Mitchell 0 1	*42 sec.*	*(Very "hard" edges)*
Mitchell 0 5	*42 sec.*	*(Plain splitting of the edges)*
Mitchell 0 10	*45 sec.*	
Mitchell 10 10	*44 sec.*	*(Still "hard" edges – formation of strange colorations along the edges)*
Mitchell 0.3 0.3	*44 sec.*	*(Default values)*
Mitchell 0.3 1	*43 sec.*	*(Similar to the filter Catmull-Rom)*
Mitchell 0.3 2	*42 sec.*	*("Glow" effect along the edges)*

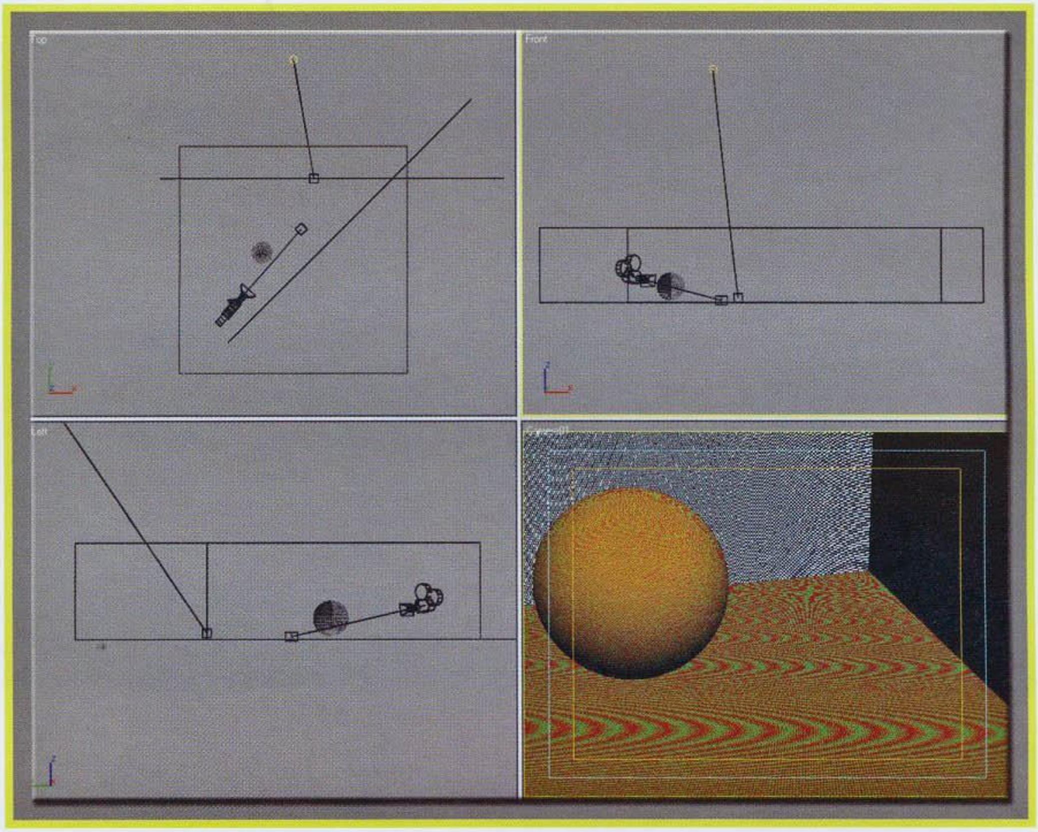
The choice of the kind of filter is very diverse. From studies on works carried out in these years, for static images the filters ***Catmull-Rom*** and ***Mitchell Netravali*** are mostly used. Because of their "sturdiness", these filters might create visual problems for very small and regular objects, such as grids or floorings (see the following). For this reason it is preferable to sometimes use softer filters, such as ***Area*** = 1.5 or ***Video***, most of all in animations.

Example 11: moirè effect

The next example shows filter behaviour when very detailed and regular textures are present, like a chess-board or a very dense pavement. Because of their complex nature, a specific problem could occur during the rendering, called "moiré effect". Very accurate filters (sharp) such as ***Mitchell Netravali*** and ***Catmull-Rom*** might even increase this annoying problem, even when setting high sampling values. Instead, smooth filters (blurry) such as ***Area***, ***Quadratic***, ***Cubic***, can reduce it.

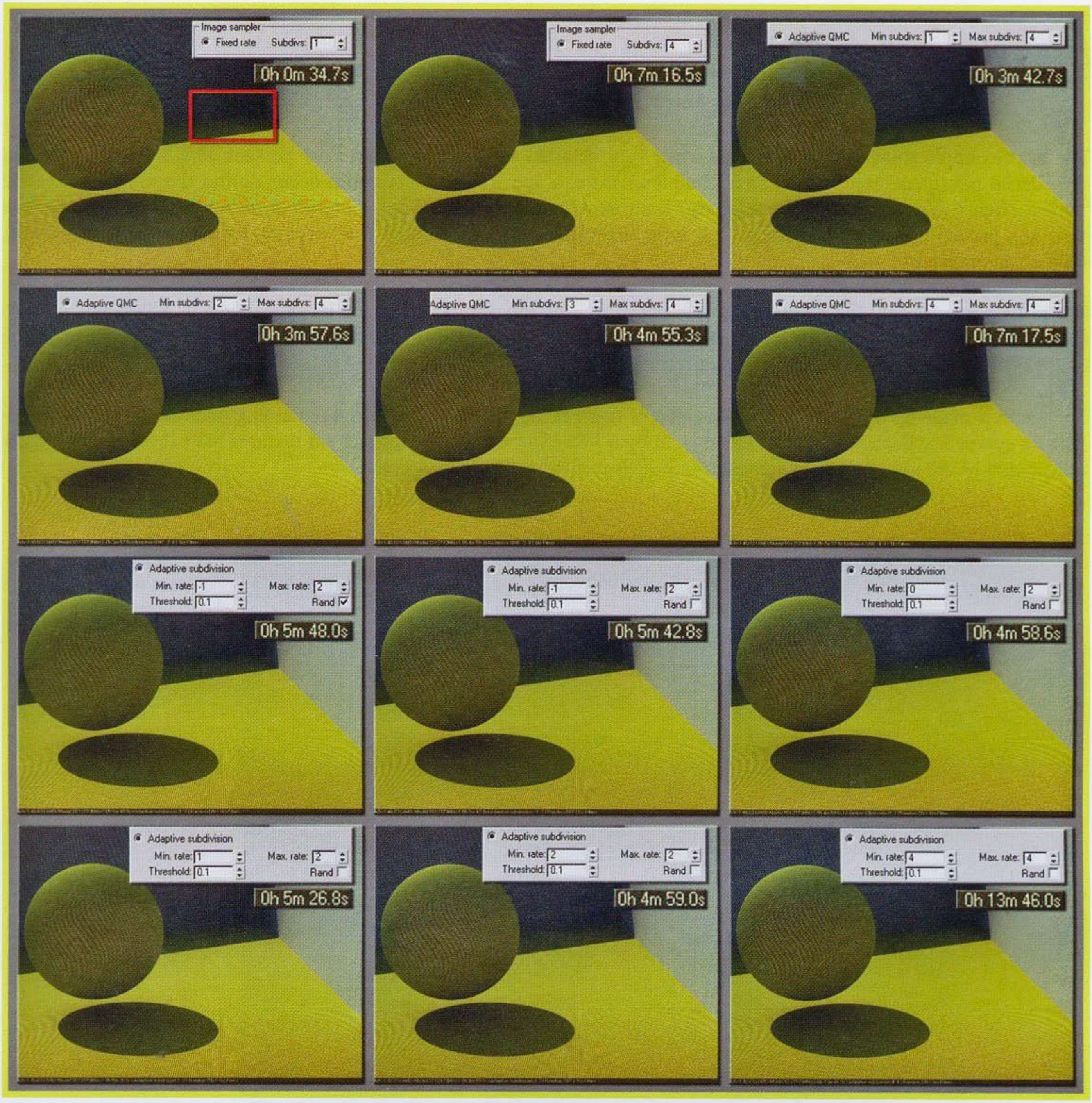
It is to be reminded that the moiré effect it is not caused by low sampling values. Its appearance is inevitable, because direct consequence of having vectorial objects turned into discrete values, in pixels. This effect may be reduced by the use of filters, but cannot be completely removed.

The scene is made up of three flat surfaces, one of these being the floor, whilst the remaining ones are the walls; at the centre a sphere has been placed. The scene is lit by a ***VRaySun+VRaySky*** system. All textures have the sampling filters activated, thus the option ***Filter maps*** present in ***VRay: Global Switches*** is active.



■ Figure 3.85
The moiré effect is even visible in 3ds Max's viewport, most of all in the pavement. The reason is the low quality of the ***AA*** of the graphic card.

Before analyzing the filters, one could ask which is the best sampling method for this scene. The model has very complex textures, but at the same time particular blurry effects are not present (***DOF, Motion Blur, Area Shadows***, etc…). Thus some tests are to be carried out to understand which is the correct ***antialiasing*** method.



■ **Figure 3.86**
Rendering of the scene with different ***antialiasing*** computation methods.
- - - DVD - - -

Figure 3.87
This image has been modified with Photoshop (desaturation, luminosity, and contrast) to mark the differences between the different tests.
--- DVD ---

Tests have been achieved in such a way to use all of the available samplers. The highest quality has been achieved with the method ***Fixed rate subdivs*** = 4, but in more than seven minutes. A new observation arises from the tests:

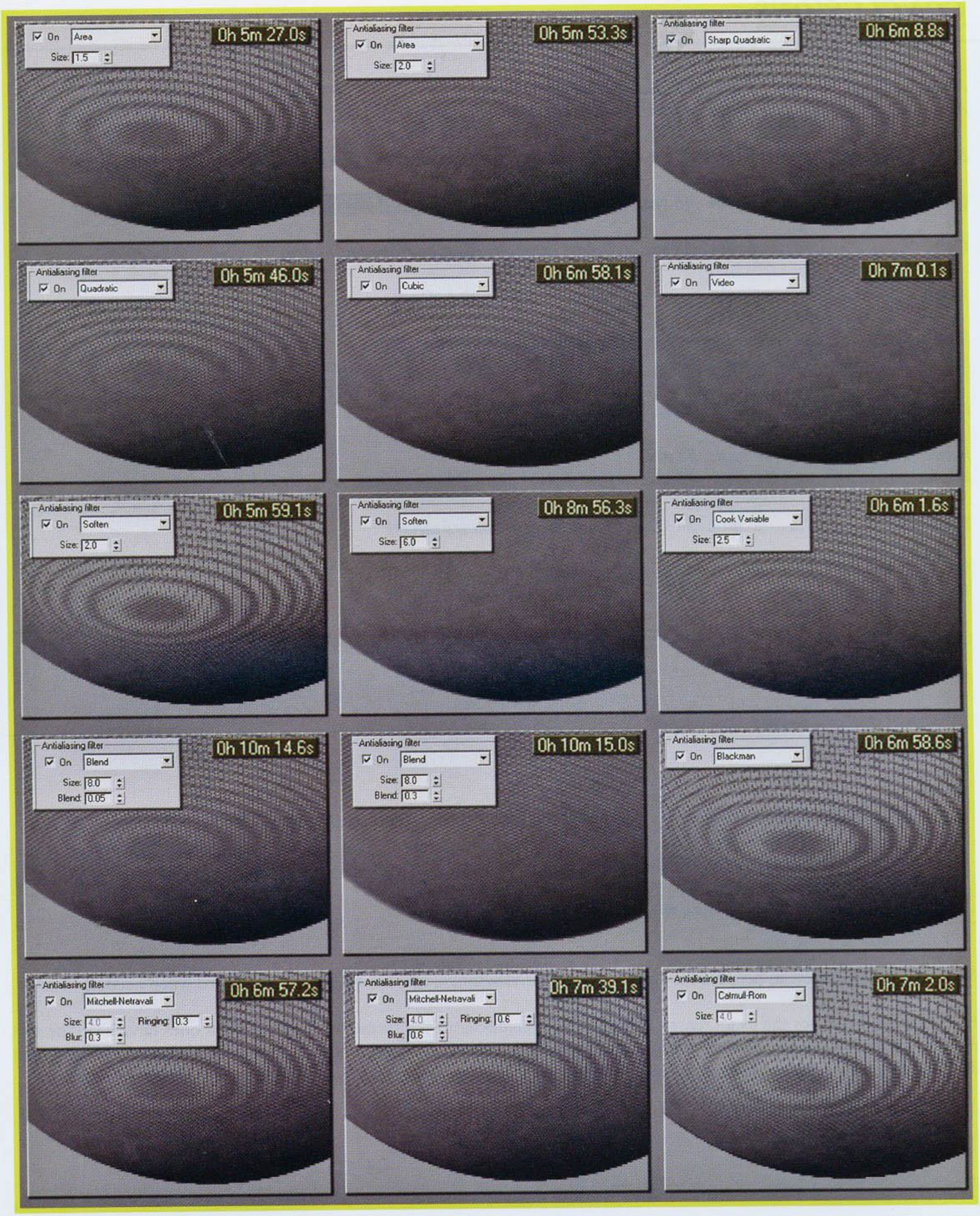
Fixed rate subdivs = 4 (7 min. 16 sec.) EQUALS ***Adaptive QMC Min subdivs*** = 4 ***Max subdivs*** = 4 (7 min. 17 sec.)

Quality and times are perfectly identical, therefore it can be stated that the same values of ***Min*** and ***Max subdivisions*** of the ***Adaptive QMC*** are equivalent to the same value applied to the ***Fixed rate subdivs***.

It has been possible to observe that, at equal quality, using the ***Adaptive subdivision Min rate*** = 2 and ***Max rate*** = 2, roughly the same quality of the ***Fixed rate*** = 4 can be obtained in just 5 min. Furthermore, deactivating the parameter ***Rand***, rendering time has been reduced, although not much, obtaining a higher quality too.

Instead, using the ***Adaptive QMC Min subdivs*** = 2 ***Max subdivs*** = 4 in four min. a good quality has been achieved, although in the shadow area some noise is always present. To remove it, the values ***Min subdivs*** = 4 ***Max subdivs*** = 4 must be reached (the same as ***Fixed rate subdivs*** = 4), a solution which requires roughly 8 min. Thus, from now on, ***Adaptive subdivs Min rate*** = 2 and ***Max rate*** = 2 will be used.

■ **Figure 3.88**
Application of the filters. - - - DVD - - -



■ **Figure 3.89**

Enlargement of the lower part of the sphere. In order to point out the moiré effect some adjustments have been carried out with Photoshop (desaturation, luminosity, and contrast). - - - DVD - - -

CONCLUSIONS

Area 1.5 ……..	***5 min. 27 sec.***	***(Moiré decreasing)***
Area 2 ……..	***5 min. 53 sec.***	***(Further lowering of moiré)***
Sharp Quadratic …….	***6 min. 8 sec.***	***(Almost complete disappearance of moiré)***
Quadratic ……………	***5 min. 46 sec.***	***(Better both in quality and in rendering time)***
Cubic ………………..	***6 min. 58 sec.***	***(Almost complete disappearance of moiré)***
Video …………………	***7 min.***	***(Presence of the moiré effect)***
Soften 2 ……………..	***5 min. 59 sec.***	***(Presence of the moiré effect)***
Soften 6 ……………..	***8 min. 56 sec.***	***(Excessive smoothing - Bloom effect)***
Cook Variable 2.5 …	***6 min. 5 sec.***	***(Very good)***
Blend 8 0.05 ….	***10 min. 15 sec.***	
Blend 8 0.3 ….	***10 min. 15 sec.***	***(The slowest one - Bloom effect)***
Blackman …………	***6 min. 58 sec.***	***(Presence of the moiré effect)***
Mitchell 0.3 0.3 …	***6 min. 57 sec.***	***(Presence of the moiré effect)***
Mitchell 0.6 0.6 …	***7 min. 39 sec.***	***(Presence of the moiré effect)***
Catmull-Rom ……..	***7 min. 2 sec.***	***(The worst one. The moiré effect has even increased)***

As can be noticed both from images and from the table, filters can be subdivided in three classes:

9. Soft filters: ***Sharp Quadratic***, ***Quadratic***, ***Cubic***, ***Video***.
10. Sharp filters: ***Blackman***, ***Mitchell***.
11. Mixed filters: ***Area***, ***Soften***, ***Blend***.

Mixed filters are called so because they allow personalized settings, and can be made soft or hard when needed.

In the previous test, for the exact reason of the intrinsic character of the scene, the most efficient ones have been the soft ones. Generally for static images without any specific problem of regular textures, the employment of ***Catmull*** and ***Mitchell*** is the best choice. Instead, in case of animations or when the moiré effect is present, soft filters are better.

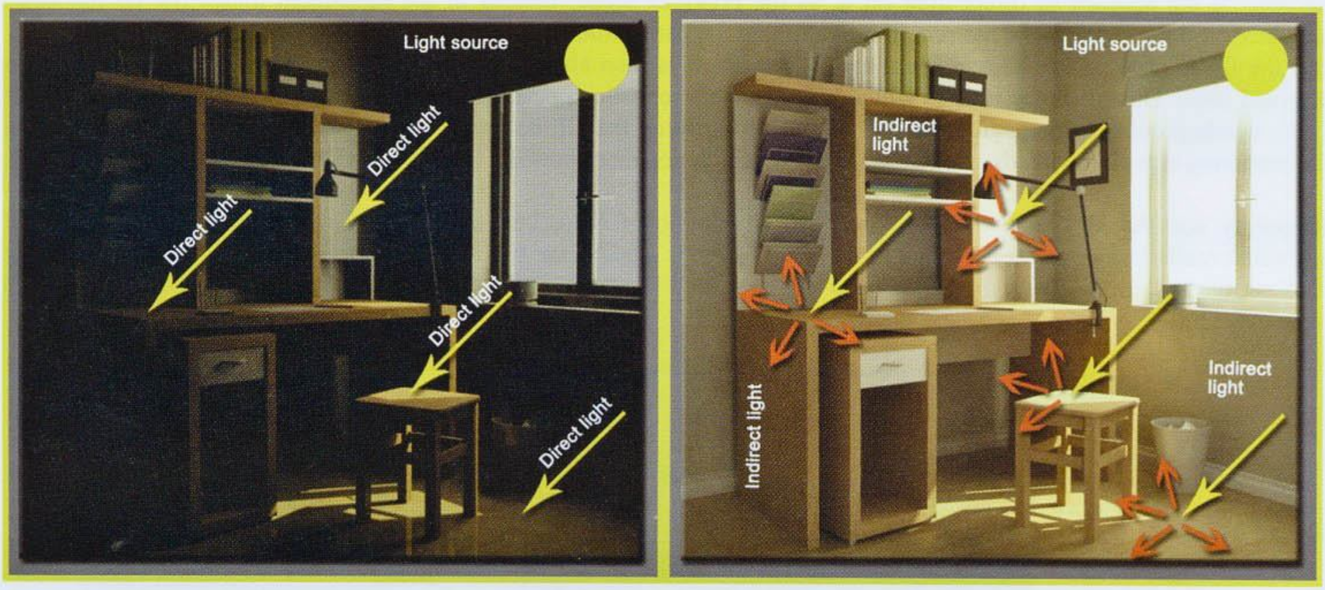
VRay: Indirect Illumination (GI)

INTRODUCTION

When one mentions **GI** (**G**lobal **I**llumination), one refers to a particular algorithm used in computer graphics. This algorithm determines both how the light hits geometric surfaces, therefore its final appearance, and the way the light tends to propagate indirectly through materials.
Radiosity, Ray-Tracing, Beam Tracing, Path Tracing, Metropolis light transport, Photon Mapping, Final gather, Irradiance Map and Light Cache are examples of algorithms involved in the calculation of Global Illumination. Some of them can even be used together.

To best understand what the GI consists of, nothing is better than examples.

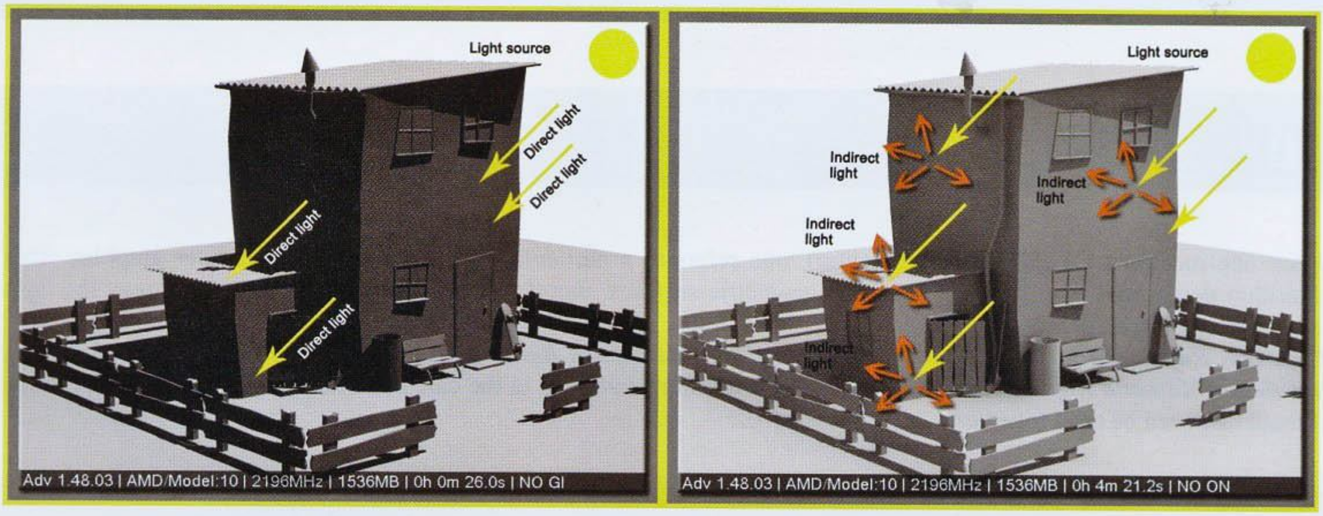
In the following scene a room's interior is to be rendered. In the left image the render without the GI is shown, while on the right the GI is activated. It is quite obvious that the second image's quality is higher, "just" because of the activation of this simple function.



■ **Figure 3.90**
Rendered scene with and without Global Illumination.

The main light source, the Sun, is a Direct Light, the typical 3ds Max source. If this term sounds new, it is advisable to carefully read the 3ds Max online guide. One thing is to be noticed. In the first image the model's polygons which are not directly exposed to light appear to be completely black. This is quite obvious, since there is no indirect illumination system allowing polygons which are not exposed to the Sun's light to be illuminated. The only way of lighting them, without using the GI, is exposing them to another light source.
When the GI is activated, *VRay* uses both direct lights hitting the polygons and the bounced light too. Once it has hit the surface, the light ray does not stop after the first collision, but goes on, rebounding like a ping-pong ball, until complete loss of energy. It is more or less what happens when a rubber ball is launched inside a room. It would be unnatural if the ball stopped after the first collision (obviously unless we consider the case of walls completely covered by instant glue!). Instead the ball goes on until its energy is completely finished. One of the most complicated aspects of fulfilling a high-photorealistic render is stating when and how to make this energy spread or stop.

This second example is even clearer than the first one in showing how Global Illumination is able to change the photorealistic appearance and the quality of a render.

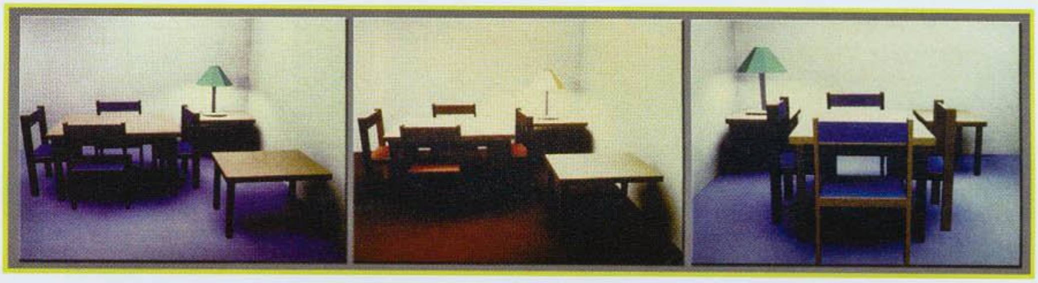


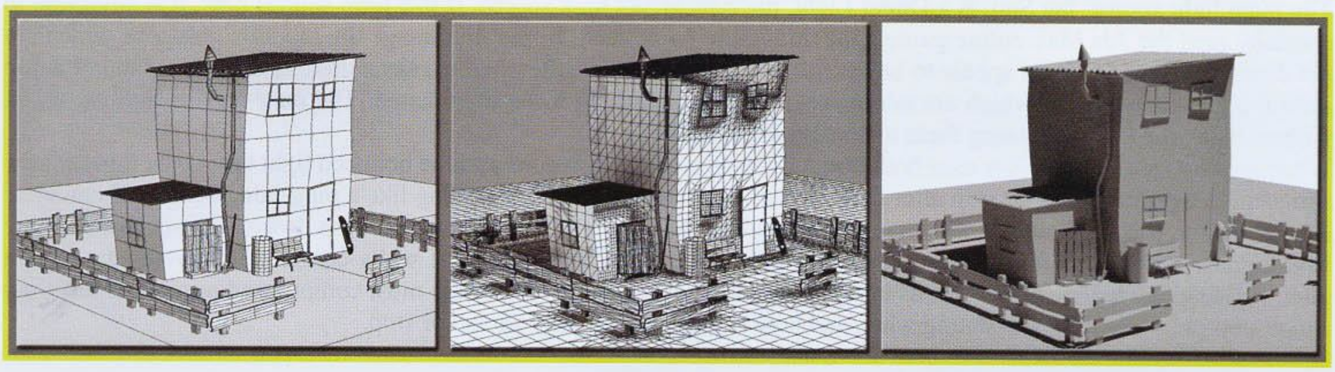
■ **Figure 3.91**
Another example of scene rendered with and without Global Illumination.

In the left-hand image the building is lit with sunlight. The side of the house that is in the shadow is completely black, because of the absence of direct light on it, as the previous scene. Instead in the right-hand image the GI has been activated, and the result is the indirect illumination on the parts which are in the shadow.
But the GI is not just an indirect illumination system. It is also able to simulate the light basing this simulation on physical laws. This gives the images which use GI such a realistic appearance. In fact the mathematical algorithm, that is the whole process of managing the formula and controlling the GI, has been created with the aim of approximating the rendering equation, already introduced in 1986, which describes how the light is transferred through the space.

Radiosity was the first algorithm for a realistic simulation of light. Less powerful than systems used nowadays, it allowed a realistic computation of illumination. The famous ***Lightscape*** was one of the first commercial software using it. A similar system has been introduced now in 3ds Max as well, but it is less used compared to new methods (much more efficient).

■ **Figure 3.92**
Three rendered images made in 1985, taken from the paper written by Michael Coben and Donald Greenberg where all the qualities of radiosity were shown. They are rendered at 512x480 using a VAX 11/780.



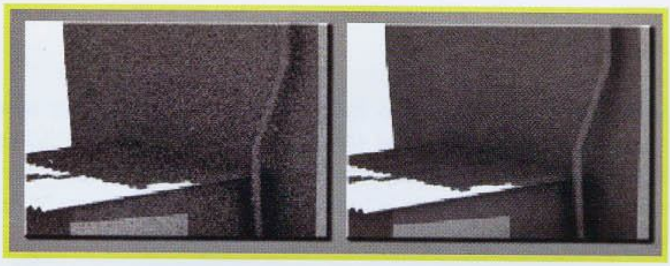


■ **Figure 3.93**
Wireframe of the scene after the subdivision process and final render both carried out with the ***Radiosity*** method in 3ds Max.

One of the most important shortcomings in ***Radiosity*** is due to the fact that every mesh, a simple cube too, must be subdivided in little pieces to be rendered. The smaller the pieces are, the higher the quality of the render. One could legitimately think this would require lots of energy in terms of hardware resources. Another limitation of ***Radiosity*** is that it is not able to execute operations such as refractions, reflections, translucence and caustics. This is due to its intrinsic nature, not allowing it to actually compute these effects. A strong point of ***Radiosity*** is that it is independent from the point of view. What does this mean? Imagine being in the exact point of the camera where the 3D model is to be observed and rendered. Now start the ***Radiosity*** process. The program computes the lighting-technique solution not only for the 3D models visible through the camera, but for all of the objects in the scene, including the ones located behind the point of view. This means that, once the ***Radiosity*** process has been carried out, if the observer's frame is changed to another position, it will not be necessary to compute the solution again: the one which has just been obtained contains all the information. All this information can be stocked inside a separate file. One should take note that if an object is moved into the scene, or its color or shader or the light is changed, the previous scene must be computed again, because of the changes. But this happens in all the other computing algorithms too.

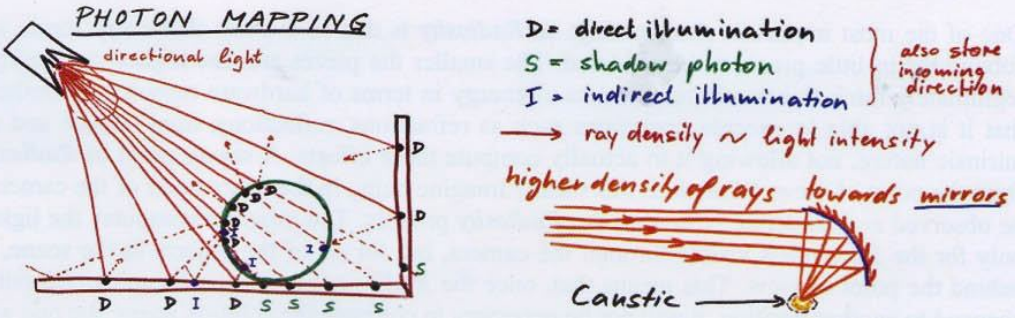
In 1986 **James T. Kaijya** introduced the **Rendering Equation**. He managed to establish how light propagates inside a 3D scene. Once again **Kaijya** suggested a method for computing an image using his formula: ***Path-Tracing***. This method, which can be considered as an extension of ***Ray-tracing***, allows one to obtain global illumination effects such as color bleeding, spread reflections, or caustics. Compared to ***Radiosity***, where the rays necessary for the calculation of illumination are projected by a luminous source, with ***Path Tracing***, rays are emitted from the observer's point of view. Each time one of these rays crosses a surface in a certain X point, from this point other rays are emitted randomly from the hemisphere determined by the normal N of the polygon in X. This process stops because a maximum number of bounces are selected by the user. If not, the number of bounces would be infinite, and the contributions more and more negligible both qualitatively and quantitatively. This computing procedure creates a very thick mesh of paths. The higher the number of paths is, the more each pixel value converges in the **Rendering Equation**'s integral computing.

Another method to solve Kaijya's equation, besides ***Path Tracing***, is ***QMC*** (**q**uasi **M**onte-**C**arlo). The main difference with ***Path Tracing*** is that GI's computing is not carried out with progressive approximations, but with buckets (these methods are to be deeply described). Like ***Path Tracing***, ***QMC*** works the opposite way to ***Radiosity***. While the latter uses luminous rays emitted by light, computes the bounces and thus the lighting solution of the whole scene independently from the point of view, ***QMC*** renders the model by emitting a certain number of rays from the observer's frame, all of which hit objects present in the scene, and deriving their characteristics such as color, specularity, reflection, refraction, translucence, etc… it is possible to render a wide number of effects this way, otherwise impossible to obtain with ***Radiosity***.
For these two last methods as well, ***Path Tracing*** and ***QMC***, weak points exist. First of all, both of them depend on the point of view. Each time the user wants to render a new portion of the scene he is obliged to compute the entire lighting again, including the areas already computed in the previous scene. Secondly, they are very slow systems, requiring extremely powerful and fast hardware. Since they need lots of samples to obtain sufficient quality and not having any type of algorithm for the approximation, the calculus needs big quantities of data and rays. To avoid noise problems in the render, a defect similar to what happens to a television with poor-quality reception, the only solution is increasing the number of rays used in the calculation of the GI, increasing the rendering time too.

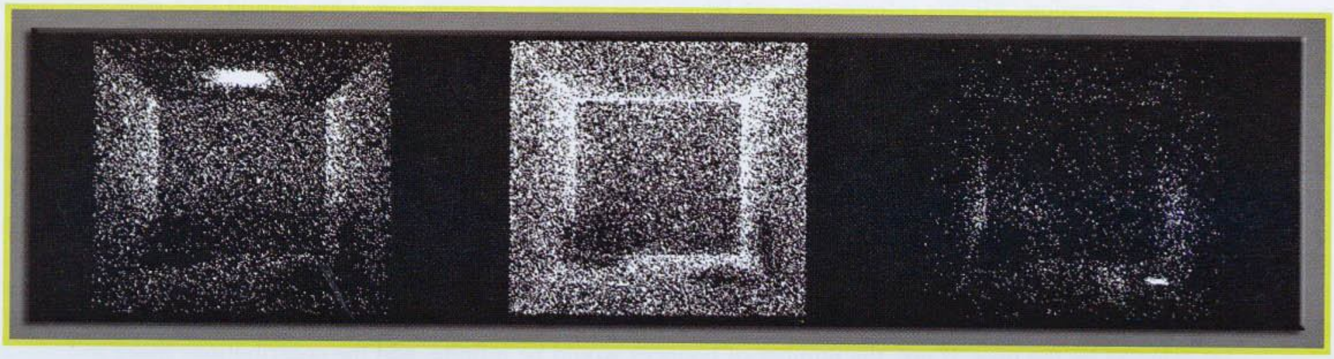


■ **Figure 3.94**
Scene rendered using different numbers of rays. On the left the numbers are very low, as can be seen by the general noise afflicting the whole render.

To solve the problem of "noise" and "high computing time", **Henrik Wann Jensen** introduces the ***Photon Mapping*** between 1995 and 1996. The ***PM*** is an algorithm extending the ***Ray-tracing*** concept and **Kaijya**'s ***Path Tracing*** idea. It can be defined as a ***Bidirectional Path tracing*** with cache (memory). In this case too, light uses natural paths that are from the source to the objects. The paths are saved in a structure called a photon map. The map contains information such as the position of the points to be hit, arrival direction and the luminous power transported.



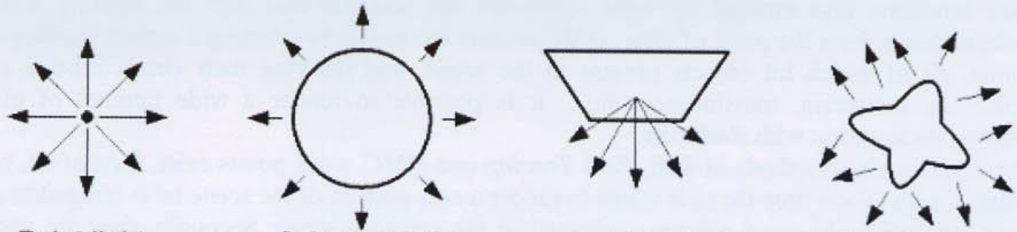
■ Figure 3.95
Henrik Wann Jensen's notes.



■ Figure 3.96
KD-tree, the map representing position, direction and power of photons. Each white spot represents one of them.

Jensen's formula is an algorithm in two steps. The first step is the launch of photons, representing the flux of energy transported from the sources in the scene; a certain number of photons are emitted by each source. They transport a quantity of power equal to that of the light source divided by the number of emitted photons.

■ Figure 3.97
Typologies of luminous sources available in the ***Photon Mapping***.

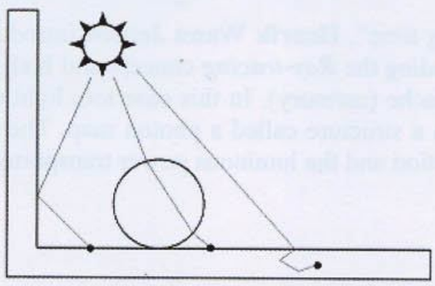


Each launched photon might be lost, meaning it might not collide with any object in the scene, or it might bump into a geometrical object. In this case there are several possibilities:

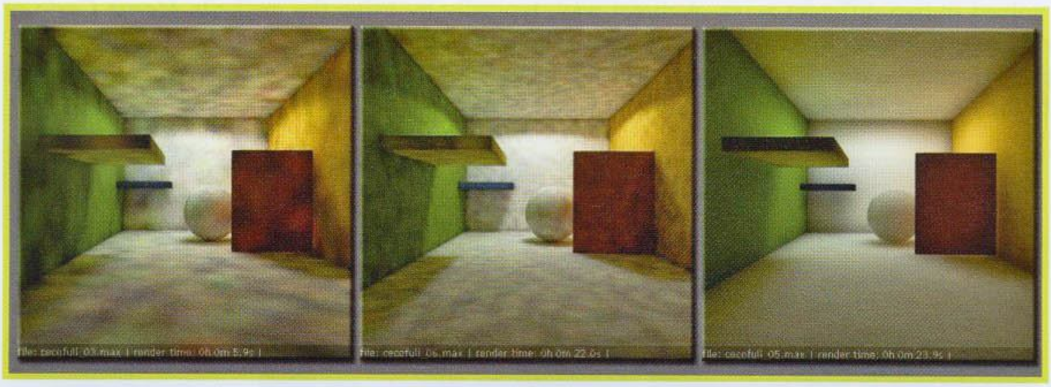
1) the photon is absorbed and saved in the map of photons;
2) the photon in diffusely reflected and its collision is saved in the map of photons;
3) the photon is reflected in a specular way.

The real nature of photons would allow them to infinitely rebound, but this would be impossible to realize in a mathematical model. If it were in this way, the first step of a photon launch would never end. Thus, at each bounce, its "history" is decided with a Russian roulette mechanism. It might be absorbed, reflected in a diffuse way, or in a specular way. In case of reflection, photon power is reduced, due to the property of diffused or specular reflection of the shaders of the objects which photons collide against.

■ Figure 3.98
Scheme of collisions between photons and geometries.

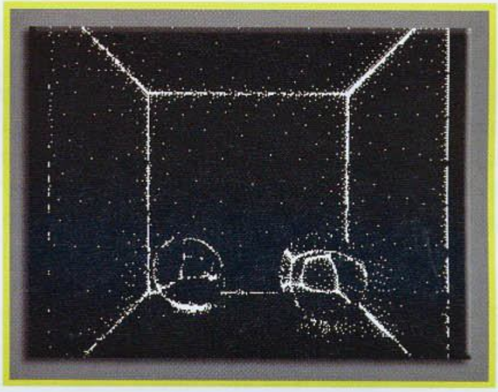


In the second step lighting is computed for a specific point; for reflections and refractions the photons have been managed as in a normal ***Ray-tracer***, whereas the computation of direct and indirect lighting intensity and of the caustics has been carried out through the density estimation, a technique born in the field of statistics.



■ **Figure 3.99**
Three renders carried out with different qualities inside VRay with the ***Photon Mapping*** technique.

To improve rendering quality, it is possible to compute the end of diffused indirect lighting with ***Final Gathering***. ***Final Gathering*** is a system which allows one to compute the lighting intensity through an estimation of the density of all the points visible from an examined point X; it allows one to sample the points around its hemisphere and then make the statistic average of them. It is an demanding procedure, that can be speeded up by using cache.



■ **Figure 3.100**
Computating points of direct lighting through ***Final Gathering***.

Besides these traditional systems for GI's computing, *VRay* introduces two of its own systems, named ***Irradiance Map*** and ***Light Cache***. Further on in the book, ***Path Tracing***, ***QMC***, the ***Photon Map***, the ***Irradiance Map*** and the ***Light Cache*** are to be widely and deeply discussed. Thus there is no need for alarm if some passages in the previous concepts were difficult to understand. As reading continues, everything will become clearer.

COMPUTING METHODS FOR THE GI

Most of the rendering engines are nowadays based on the rendering equation introduced by ***James T. Kaijya***. This equation describes the propagation of light within a 3D scene. Moreover in his report he proposed a new method for the rendering computation, using ***Path Tracing***. This system was based on a probabilistic method called ***Monte Carlo***.
This equation is just an approximation of the famous Maxwell equations for electromagnetism (1860). In fact the original model developed by Kaijya was not based on optical phenomena, but on pure geometry. Thus it could not simulate phenomena such as diffraction, interference and polarization. For the same reason the use of ***Ray-tracing*** is very suitable.

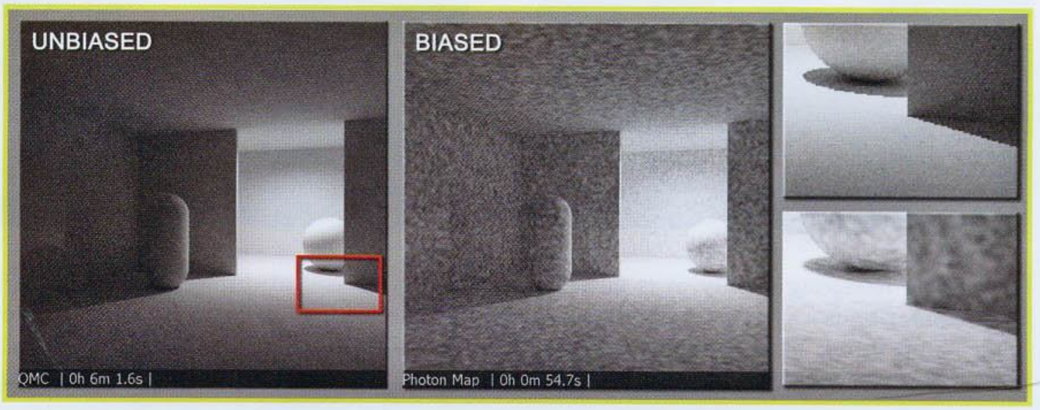
$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_\Omega f_r(x,\vec{w}',\vec{w})L_i(x,\vec{w}')(\vec{w}'\cdot\vec{n})d\vec{w}'$$

There is just one ***Kaijya*** equation. Through the years new software and methods have evolved with the specific purpose of solving it in the best way possible. It is correct to think that two different softwares solving this equation with high accuracy would give the same image, but this is true only in theory. In fact practically different engines often delete or vary some parts, giving back different images.

1 - Exact and Approximated calculation methods

Kaijya's equation can be solved in two different ways. A direct way, typically by using the methods ***Monte Carlo*** or ***quasi-Monte Carlo***, or with approximation systems, such as ***Radiosity*** or ***Photon Mapping***. What is the difference? Essentially they differ for the quality/time ratio and the kind of artefacts created. If one renders the same scene with ***QMC*** before, and then with the ***Photon Map***, the result is the following:

■ Figure 3.101
The same scene rendered with the two methods. The first one, unbiased, is very detailed, but contains some noise, while the second one is much cleaner, but it contains stains due to the approximations, one of the biggest problems of the biased methods. To be noticed: rendering-time, much better in the approximation method.



In the left image a direct method of the rendering has been used, thanks to the ***QMC*** method. It is a very accurate method for the GI computation. GI quality and shadow precision are very high. Although, as one might imagine, this kind of solution can cause some problems. First of all the time required. Even with low values of ***QMC***, times are extremely high. The second problem is the presence of noise, due to the low number of projected rays. In this test rays have been kept intentionally low, in order to avoid excessively long rendering times. Another problem is that this method depends on the point of view. Not one bit of information has been saved during the Global Illumination computation. All data created by the ***QMC*** is used for the rendering and, once finished, all information is deleted.
Instead, using one of the numerous "approximate" systems of the GI's computing (in this case the ***Photon Map***), the rendering time has been drastically reduced, obtaining however a GI with "stains" or "blotches". This technique provides other benefits. First of all a general regularity in image quality and the absence of problems such as noise. Moreover, it is possible to save the GI computation, using it again in successive views, thus saving time. Finally it is to be noticed how the render appears lighter with the ***Photon Map***, because of the higher number of bounces. This subject is to be considered later in this book.

EXACT METHODS (unbiased)

ADVANTAGES

- They produce high-quality and very accurate results.
- The only artefact produced is general noise.
- GI setting is managed with the use of very few parameters.
- Low use of the RAM.

DISADVANTAGE

- Unbiased methods are non-adaptive and very slow in case one wants to obtain high-quality images without noise.
- Some effects cannot be computed, for example caustics generated by a pointlike light reflected on a mirror.
- Typically, exact methods work directly on the final image. The GI solution cannot be saved and thus reused.

EXAMPLES

- ***Path tracing*** (***brute-force*** in some render engines, or QMC).
- ***Bi-directional Path Tracing***.
- ***Metropolis light Transport***.

APPROXIMATED METHODS (biased)

ADVANTAGES

- High adaptability, therefore a higher flexibility in spite of a possible decrease in quality.
- They are able to compute effects not reproducible with direct methods.
- For some methods, GI computation can be saved and reused.

DISADVANTAGES

- Less accurate compared to direct methods. Possible presence of mistakes which can however be reduced to a size such to not be noticed.
- Presence of problems such as stains or mistakes in the calculation of the GI.
- Many more commands and settings to be known and adapted.

EXAMPLES

- ***Photon Mapping***.
- ***Irradiance Map*** in *VRay*.
- ***Radiosity***.
- ***Light Cache*** in *VRay*.

HYBRID METHODS: use of exact methods for particular effects and approximated methods for other ones.

ADVANTAGES

- Combining the qualities of both methods, among which a particular computation speed and an excellent rendering quality stand out.

DISADVANTAGES

- It can be difficult to adjust settings.

EXAMPLES

- In Mental Ray: ***Final Gathering*** with values ***Min/Max radius*** = 0/0 + ***Photon Mapping***.
- In 3ds Max: ***Light Tracer*** with ***Min/Max rate*** 0/0 + ***Radiosity***.
- In *VRay*: ***Irradiance Map + QMC***.
- In *VRay*: ***QMC + Light Cache***.
- In *VRay*: ***QMC + Photon Map***.

2 - Shooting methods vs. Gathering methods

SHOOTING METHODS

This classification derives from how the GI is computed. We talk about **Shooting** methods when the rays used for the computation of the GI are emitted by a light source through space. Vice versa, if rays are emitted from the observer's frame, we talk about **Gathering**. Both of them can be exact or approximated methods.

ADVANTAGES

- They allow the simulation of effects such as caustics, for example.

DISADVANTAGES

- As these methods do not depend on the point of view, often it might happen that one computes areas not considered in the view. Imagine rendering a labyrinth with the GI activated. If one uses the ***PM*** photons might hit walls not visible with the camera, with loss of time and resources.
- Image quality is higher nearer the light source. Further away areas are dirtier and present more artefacts, as it is when the ***Photon Map*** is used.
- Some types of light sources cannot be simulated, such as self-lighting objects or ***skylight***. Moreover luminous sources which are not physically correct may give more problems when computed.

EXAMPLES

- ***Photon Mapping*** (approximated).
- ***Particle Tracing*** (approximated).
- ***Light Tracing*** (exact).
- ***Radiosity*** (approximated).

GATHERING METHODS

ADVANTAGES

- They produce computations for the part of the image visualized only. Normally it is the most efficient method for static images.
- They are able to produce very accurate images for all the portions of the rendered model.
- They are able to simulate luminous effects such as self-lighting objects, ***skylight*** and physically incorrect images.

DISADVANTAGES

- Effects such as caustics generated by pointlike lights or little ***Area Lights*** are very difficult and sometimes impossible to simulate.

EXAMPLES

- ***Path Tracing*** (exact).
- ***Irradiance Map*** *VRay* or final gathering in Mental Ray (approximated).
- ***QMC*** in *VRay*.

HYBRID METHODS

They combine both Shooting and Gathering methods, collecting advantages and disadvantages. As seen previously, both exact and approximated methods can be used.

ADVANTAGES

- Essentially, it is possible to simulate every kind of effect and render any luminous source.

DISADVANTAGES

- It is more difficult to set them.

EXAMPLES

- In Mental Ray ***Final Gathering + Photon Mapping*** (approximated).
- In *VRay* ***Irradiance Map/QMC GI + Photon Map*** (approximated).
- ***Bi-directional Path Tracing*** and ***Metropolis Transport*** (exact).

3 - Approximate Methods: View dependent vs. View Independent

Approximate methods can be in turn subdivided in two other groups: View dependent and View independent. They are defined View dependent when the GI depends on the view; instead in View independent methods the GI does not depend on the view.

SHOOTING METHODS (view independent)

ADVANTAGES

- They compute the GI also in areas which are not visible from cameras. It is useful for Fly-through animations.

DISADVANTAGES

- GI quality is typically lower than in other methods, producing stains and detail-loss. It is nevertheless possible to gain precise and detailed solutions, at the expense of an increase in the rendering time and in the memory used.
- Regions far from sources produce low quality.

EXAMPLES

- ***Photon mapping***.
- ***Radiosity***.

GATHERING METHODS

Gathering methods and some types of hybrids allow both dependent and independent solutions.

VIEW DEPENDENT SOLUTIONS

ADVANTAGES

- Only the visible parts of the scene are rendered, saving time and resources.
- They produce high-quality images, preserving details.
- They require less memory compared to View independent systems, because of the lesser amount of data stocked.

DISADVANTAGES

- It is necessary to make a new GI computation for new views. In some implementations, for static images, the solution can be partially reused.

EXAMPLES

- Irradiance caching (*VRay*, Mental ray, finalRender, Brazil r/s, 3ds Max light tracer).

VIEW INDEPENDENT SOLUTIONS

ADVANTAGES

- The need only be computed just once.

DISADVANTAGES

- Non-visible parts of the scene are computed as well.

- Usually geometries are reduced to triangular or square meshes. Therefore it is not possible to render procedural geometries.
- A solution with high resolution requires a lot of memory.
- Only the solution of the GI which is part of the diffused channel is computed. Effects such as Glossy reflections must be computed separately, without possibility of caching.

EXAMPLES

- Some ***Radiosity*** methods.

HYBRID METHODS

In some cases the employment of different combinations between View independent and View dependent systems.

EXAMPLES

- In *VRay* ***Photon Mapping*** and ***Irradiance Map***.
- In Mental ray ***Photon Mapping*** and ***Final Gather***.
- In 3ds Max ***Radiosity*** and ***Light Tracer***.

VRay supported methods

VRay supports different methods for GI computation: exact, approximate, Shooting and Gathering. According to the kind of job and the scene, some methods are to be preferred against others.

EXACT METHODS

VRay supports two types of exact methods: the ***QMC*** and the ***PPT*** (**P**rogressive **P**ath **T**racing). The ***QMC*** works in a traditional way, by using a system based on buckets, which consists of the reconstruction of the image zone by zone, usually squared. The Path Tracing rebuilds the render "little by little", refining the image gradually.

APPROXIMATE METHODS

Irradiance Map, ***Light Cache***, and ***Photon Map***, or rather the other methods of *VRay*, are approximate methods.

SHOOTING METHODS

The ***Photon Map*** alone is to be considered a shooting method in *VRay*.

GATHERING METHODS

Irradiance Map, ***Light Cache*** and ***QMC GI*** are Gathering methods.

HYBRID METHODS

VRay can use different kinds of engines for the primary and secondary bounces. The primary bounces, computed by a first engine of the GI, are the first bounces the light is subjected to just after the emission. After the first bounce GI's computation is managed by a second engine, which computes the following rebounds. Thus, it is possible to use two different computation methods, one for the primary bounces, and one for the secondary ones.
Now a question might arise. How can one know which is the best solution for his job, among all these possibilities? Actually there is no rigorous rule. For example, a room can be rendered with the method ***QMC+QMC***, ***QMC+LC***, or ***IM+LC***, or ***IM+PM*** too. All these pairs might give the same result, but following different procedures and paths, each one with its advantages and disadvantages.

INDIRECT ILLUMINATION (GI) IN VRAY

VRay contains many methods for GI computation:

- ***QMC***, also named ***Direct computation*** or ***Brute force***.
- ***Irradiance Map***.
- ***Photon Map***.
- ***Light Cache***.

DIRECT COMPUTATION (QMC GI)

It is the easiest method to set. Few parameters determine the quality of the Global Illumination. Instead high rendering times are to be expected.

ADVANTAGES

- It preserves all the details, such as small shadows of indirect light or complicated geometries. Thanks to the ***QMC*** very detailed and accurate images can be obtained.
- It is possible to solve flickering problems using the ***Direct computation*** (in other words the presence of "vibrating" pixels which can ruin the playback of the video).
- It uses little RAM.
- When using ***Motion blur***, indirect light is correctly computed.

DISADVANTAGES

- It is a very slow method, most of all for complex scenes, such as indoor ones.
- The real problem for this method is the presence of noise, which involves the whole render, most of all in the shadowy areas. The only way to set it up is by increasing the values in the settings, which in turn increases computation times.

IRRADIANCE MAP (IM)

It is a method based on the ***Irradiance Caching***. In this case the GI is computed in strategic points of the scene only. Usually these are in areas requiring more attention, like mesh areas. For example between very near polygons or very detailed geometrical shapes. It is up to the user to decide how many points must to be considered, where to locate them, and how their interpolation has to be computed. The ***IM*** requires many more setup parameters to be adjusted compared to the ***QMC***, and their correct knowledge, practice and experience can lead to high-quality results in a short time too.

ADVANTAGES

- It is very quick to compute compared to the ***QMC GI***, especially on scenes containing models with even surfaces.
- Noise is practically absent. This method's main problem is the appearance of "stains" in certain areas, due to low settings.
- The ***IM*** can be saved in an apposite cache and reused when necessary, for instance in ***Fly-through*** animations. In this case it is to be remarked that the position of every object must be kept the same, for the color and lights intensity as well. The only objects which can be moved are, obviously, cameras. This idea will be explained up ahead.
- One can use the ***IM*** for accelerating the computation of the ***VRayShadows*** produced by *VRay* lights. Usually ***VRayShadows*** by default use direct methods for shadow computation and not approximated ones, for a better quality. This computation can become very long when many ***VRayLights*** are present. In this case the ***IM*** can be useful, enabling one to speed up shadow computation.

DISADVANTAGES

- Some details, such as small shadows generated by indirect light may not be rendered correctly, due to the interpolation of the GI.
- During animations, when using low setting values, some flickering effects on the animation can occur.
- It requires much more memory than the ***QMC GI***.
- Indirect illumination is not correctly computed through objects to where ***Motion blur*** is applied. But often this defect is not evident enough to ruin the overall animation quality.

PHOTON MAP (PM)

This method computes the GI by emitting, from light sources, rays which then proceed to bounce within the scene. It is very useful for interior and/or semi-interior scenes, with many lights or little opening. Usually the ***PM*** does not produce satisfying results if used on its own. Instead, it is very useful if employed together with the ***QMC GI*** or the ***IM***. In fact, by combining the speed and the original approximation of the ***PM*** with the previous methods, one is able to obtain excellent quality images in relatively short times.

ADVANTAGES

- The ***PM*** produces a very quick computation of the GI, even if approximate and not so detailed.
- The ***PM*** can be saved and reused for speeding up the rendering of new views of the same scene and for ***Fly-through*** animations.
- The ***PM*** does not depend on the view, and, once computed, it is always available, whichever the zone of the scene to be rendered.

DISADVANTAGES

- It is not worth using the ***PM*** as engine for primary bounces because of its lack of precision, most of all for very detailed scenes.
- It requires a lot of memory.
- The ***PM*** needs real light sources for producing the GI. Self-lighting objects and environmental light (***skylight***) are not able to produce photons, therefore neither the GI.

Light Cache (LC)

Light Cache is a method for the approximate computation of the GI. It is basically very similar to ***Photon Map***, but has managed to keep only its merits, solving the faults. The ***LC*** is created by emitting the rays from the observer's frame towards the scene. The ***LC*** is an universal method for GI computation: it can be used both for indoor and outdoor scenes, and both for primary and secondary bounces, together with the ***IM*** and the ***QMC GI***.

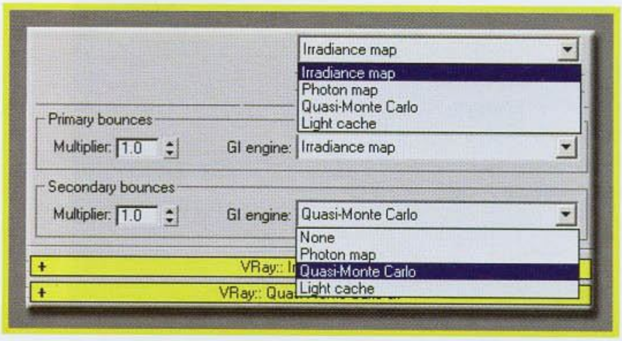
ADVANTAGES

- ***LC*** is very simple to set. Only rays deriving from the camera are used, contrarily to what happens in the ***Photon Map***, which must compute each single light source as well. In fact with the ***PM***, it is necessary to set the number of photons for each source. This operation is pointless in the ***LC***.
- ***LC*** works with any object able to emit light, including self-lighting objects, non-photometric lights and ***skylights***.
- ***LC*** produces correct results along corners in a room and near little objects. Instead, ***PM*** is not extremely efficient in corners, creating particularly dark areas.
- ***LC*** can be visualized as a preview during its computation, enabling a direct feed-back for the user on what ***VRay*** is computing.

DISADVANTAGES

- Like the ***IM***, ***LC*** too depends on the point of view, and the GI is computed for the set point of view only.
- ***LC*** does not produce excellent results when Bump effects are present. For this reason it is better use it with the ***IM*** or the ***QMC GI*** when detailed Bump effects are present.

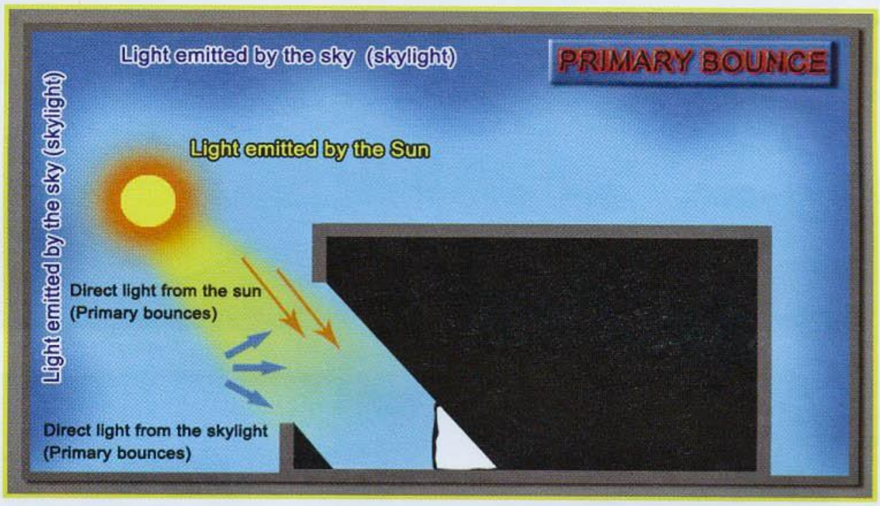
Among these four methods, which is the best one to use and for which kind of scenes it is better to use a method rather than another one? The answer is not universal, even for the same scene and the same lighting. Some methods are more useful than others, if we talk about static, dynamic and fixed renders. It all depends on what one wants to obtain.



■ **Figure 3.102**
Panel showing all the GI computation methods present in VRay.

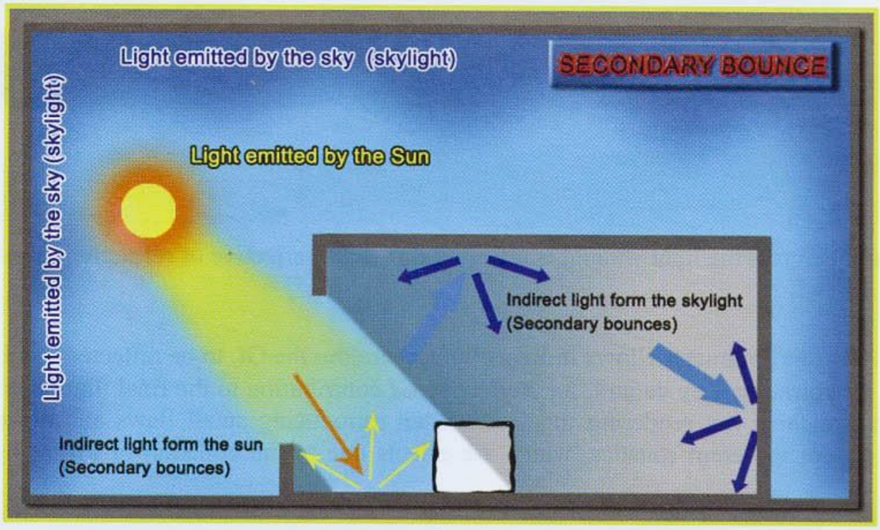
In the previous image the panel managing the GI in *VRay* is shown. It is subdivided in two sections. ***Primary bounces*** and ***Secondary bounces***.

Primary bounces represent the first step in light diffusion. Once emitted from the light source, light travels through space. When it encounters in an object, is it involved in various processes: diffusion, refraction, reflection etc… In the drop-down menu of primary bounces it is possible to choose the GI engine. It is possible to choose between the four methods available in *VRay*. In the next scene one can notice how both the sun and the sky contribute to the scene's lighting. The sun is easily reproducible in 3ds Max thanks to a normal direct light, while in *VRay* a special tool is available for the sky: ***VRay Environment***, to be found in the Renderer panel of *VRay*. Further on it will be explained in detail. For the moment, just imagine it as a way of simulating the sky.



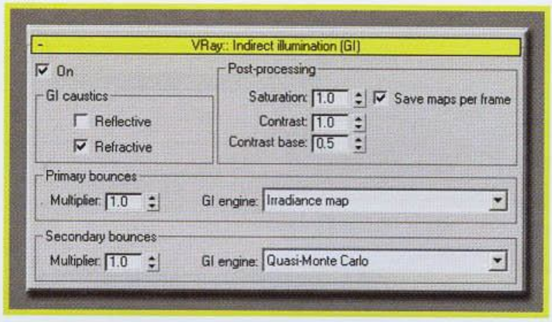
■ **Figure 3.103**
Scheme representing primary bounces.

Once the rays have collided once, *VRay* continues the computation of the path followed by the luminous flow thanks to a second rendering engine. In this case three among the four computation methods can be chosen.



■ **Figure 3.104**
Scheme representing secondary bounces.

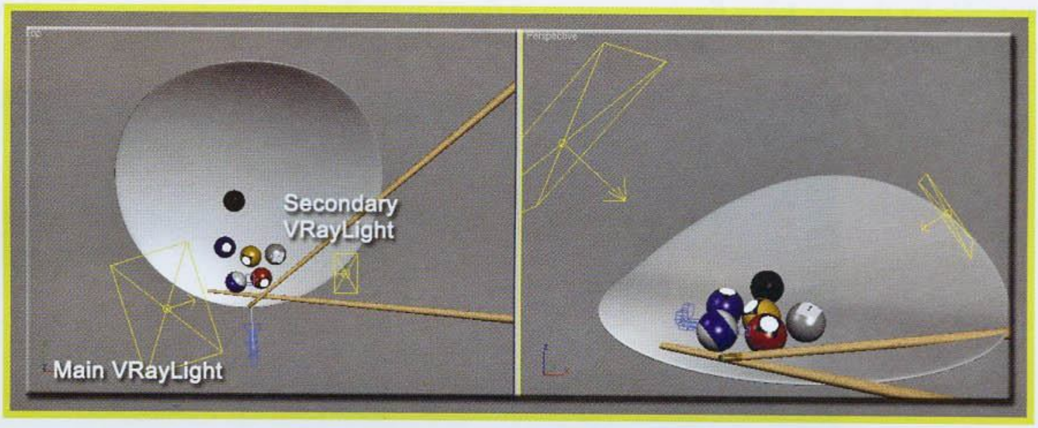
It is possible to light up parts of the scene not directly lit by the sources. *VRay* allows one to choose the system to be used for primary and secondary bounces. As one can see by the parameters present in the rollout ***VRay: Indirect illumination***, it is possible to set their intensity, setting the ***Multiplier***. So, it is possible to control light as never before!



■ **Figure 3.105**
The panel ***VRay: Indirect illumination (GI).***

PARAMETERS

on - it activates and deactivates the GI. When deactivated, *VRay* uses only the primary bounces for the scene's illumination.



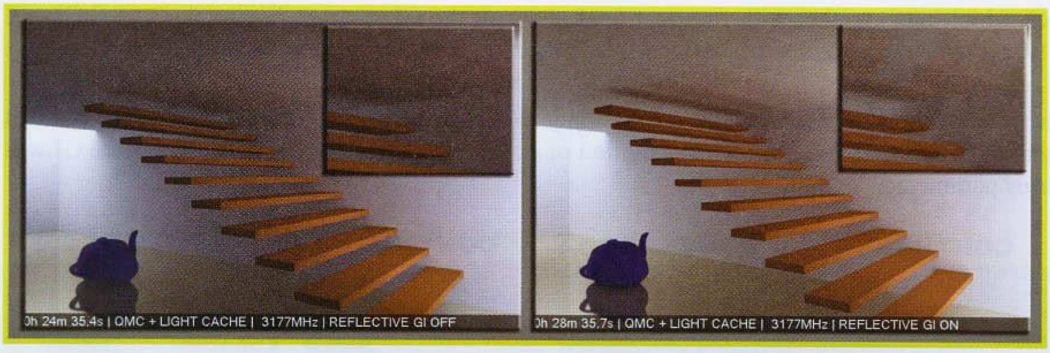
■ **Figure 3.106**
Bird-eye view and prospective view of the scene. It is composed by several snooker balls and a supporting plane. Illumination is generated by two ***VRayLights*** and by the ***Environment***.

■ **Figure 3.107**
On the left image the scene is illuminated by the two ***VRayLights***, because the ***GI*** is deactivated, while on the right one the ***skylight*** effect is added. Times are considerably increased, but so is image quality and photorealism too.

GI caustics

GI caustics represents and controls the light when it is reflected diffusely, reflective or refractive specularly.

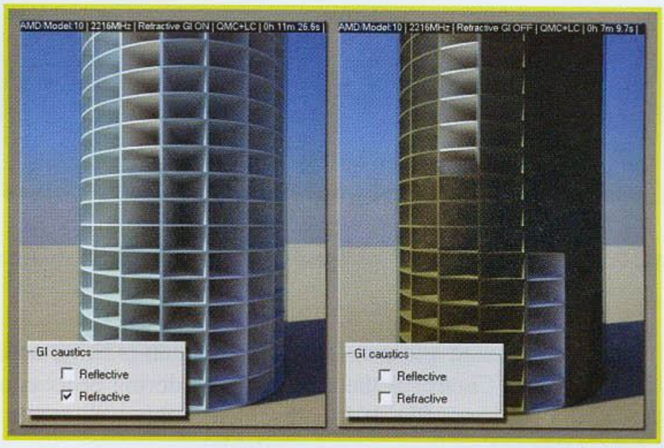
Reflective GI caustic - this parameter allows **indirect** light, therefore the GI, to be reflected by objects like mirrors and reflective surfaces. It is turned off by default, for ***Reflective GI*** contribution to the final illumination of the scene is usually low, other than stretching out rendering times and often introducing small flaws and noise. To be noticed: ***Reflective GI caustics*** do not represent caustics, which are effects due to the collision of **direct** light onto specular objects.



■ **Figure 3.108**
Scene rendered with the ***QMC+LC*** method with 25 ***Subdivs*** for ***QMC*** and 1,500 for the ***LC***. The room is lit by the ***skylight*** only. In both scenes the parameters are the same, except for the activation of the ***Reflective GI caustic*** parameter. The activation of this parameter would have been negligible with a completely opaque floor.

In the test shown two things must be noticed. The scene appears much more luminous by activating the ***Reflective GI caustic*** parameter. This happens because the floor is a colored mirror, and reflects the indirect light. Therefore in this case indirect illumination affects global illumination considerably, because the floor has a wide extension. In fact, as marked by the ceiling area, the reflection of indirect light produces new light, which in turn produces visible shadows on the ceiling. A real example of the GI reflection phenomenon might be the illumination due to a water flow. The solar light bouncing on the river's surface, lights the underside of a bridge. ***Reflective GI caustic***, which should in theory always be activated can produce artefacts. This happens because its computation requires many samples. If the number of ***Subdivision*** is not enough, some colored dots might appear in some areas of the render instead of the ***Reflective GI***. In any case, the best advice is to always keep it turned off, as its contribution of indirect illumination to the GI of the entire scene is often hardly noticeable. The second effect due to the activation of the parameter ***Reflective GI caustic*** is the light brown coloring of the whole scene, due to the floor reflection, in slang known as **Color bleeding**.

Refractive GI caustic - the activation of this parameter allows **indirect** light to pass through transparent objects. Be aware that ***Refractive GI caustic*** and caustics are not the same thing. The latter represent how the direct light passes through matter, whereas ***Refractive GI caustics*** represent how indirect light does the same.
Refractive GI caustic is to be kept active at all times; in fact it is turned on by default. In this way, for example, that ***skylight*** or any other light are able to pass through transparent surfaces, such as windows or a glass.



■ **Figure 3.109**
Exterior image rendered with ***VRaySun+VRaySky*** of VRay 1.49.03. ***QMC+LC***. The GI does not pass through glasses if ***Refractive GI caustic*** is deactivated, while the direct rays do it. ***Refractive GI caustics*** affects indirect light only. Deactivating it, the rooms of the building are in any case lit by the direct sun light, but not by indirect light, which is not able to filter through the transparent glass. Only if the ***Refractive GI*** caustics are activated the GI is able to pass through the glass windows.

Post-processing

The following series of parameters allows one to modify indirect illumination, meaning the GI alone before the final rendering calculation. As will be seen shortly, once ***IM, LC*** or ***PM*** has been saved, ***VRay*** allows one to modify contrast and saturation without having to re-calculate the whole GI. By changing the ***post-processing*** parameters it is possible to alter the GI, although the default values are the only ones which assure physically accurate results. The purpose of these parameters is to give the user greater control over the GI. Technically speaking, these parameters permit GI corrections which are typical of image processing programs.

Saturation - this parameter controls GI saturation. If the value is set to 0.0, this means that Global Illumination does not have any color, thus eliminating any type of color bleeding effect. Value 1.0 means that GI colors will remain unaltered. Values greater than 1 will increase contrast of GI and color bleeding.

Contrast - this parameter, which acts together with ***Contrast base***, controls the GI contrast. With value at 0.0 GI will not have any contrast. At 1.0 GI will remain unaltered. Values greater than 1.0 will increase contrast.

Contrast base - this parameter controls the quantity of contrast applied by the ***Contrast*** parameter. In other words it fixes the GI value which will not be modified during the calculation of contrast. The greater the value, the more GI will be influenced by the ***Contrast*** parameter.

■ **Figure 3.110**
Summary scheme for ***Contrast*** and ***Contrast Base*** parameters. The image which is highlighted in purple shows a render with default parameters.

Save maps per frame - if this activated and the ***Auto save*** function is being used, *VRay* saves GI maps (***Irradiance Map***, ***Photon Map***, ***caustics and Light Cache***) at the end of the rendering process of each frame. Take note that *VRay* always saves the maps overwriting the same file. If it is disabled, *VRay* will create the various maps once only, at the end of rendering (from ***v1.5 RC3*** this option is no longer available).

Primary bounces

Multiplier - this parameter fixes the amount of direct light which contributes in illuminating the scene. Value 1.0 will produce images which are physically correct and accurate. For correct results, it is advisable to leave ***Multiplier*** unaltered. Inside *VRay* there are many other options which allow one to change the amount of light within a given scene. Use this ***multiplier*** as a last resort.

Primary GI engine - with this pull-down menu, it is possible to specify which of the four available methods in *VRay* to use for the calculation of ***Primary diffuse bounces***.

- ***Irradiance Map*** (***IM***).
- ***Global Photon Map*** (***PM***).
- ***quasi-Monte Carlo*** (***QMC GI***).
- ***Light Cache*** (***LC***).

Secondary bounces

Multiplier - this parameter fixes the amount of indirect light which contributes in illuminating the scene. Value 1.0 will produce images which are physically correct and accurate. Values lower than 1 will decrease the amount of indirect light. For correct results, it is advisable to leave ***Multiplier*** unaltered. Inside *VRay* there are many other options which allow one to change the amount of light within a given scene. Use ***Multiplier*** as a last resort.

Secondary GI engine - with this pull-down menu, it is possible to specify which of the three available methods in *VRay* to use for the calculation of ***Secondary diffuse bounces***.

- ***Global Photon Map*** (***PM***).
- ***quasi-Monte Carlo*** (***QMC GI***).
- ***Light Cache*** (***LC***).

VRay can make use of many different methods for calculating both primary and Secondary bounces. Each one has its advantages and defects. The next step is to understand which of these it is best to use, according to the scene.

ASSOCIATING THE METHODS

With the choice of four rendering engines and two types of bounces, a beginner could feel a little intimidated.
Which of the rendering engines to use and in what way should they be associated with ***Primary*** and ***Secondary diffuse bounces***? First of all, one has to identify the types of rendering which are producible within a 3d software (animation, static rendering, etc…). It will then be necessary to assign the relative ***GI engines*** to the ***Primary*** and ***Secondary bounces***, having understood the reason behind this choice. The pairs highlighted in yellow are the ones that allow one to obtain images of the highest quality, without considering rendering time. The ones highlighted in green are the best solutions in terms of quality/time ratio.

1. INDOOR ANIMATION

- Objects and moving lights with changes in the color and geometry (DYNAMIC ANIMATION):

1.1. ***QMC+QMC*** - NOT RECOMMENDED, very slow although it is the most precise. High noise.
1.2. ***QMC + LC*** - decent speed, high precision, good and well-lit illumination, low noise.
1.3. ***IM+QMC*** - decent speed, good precision, possible presence of blotches in the GI. Flickering.
1.4. ***IM+LC*** - fast, good precision, correct and luminous illumination, very few artefacts, flickering.
1.5. ***IM+PM*** - fast, low precision in details, long setup, possible presence of spots, flickering

- With only the camera in movement (STATIC FLY-THROUGH ANIMATION):

1.6. ***QMC+QMC*** - NOT RECOMMENDED, the slowest, maximum precision, high noise, GI can not be saved.
1.7. ***QMC+LC*** - reasonably fast considering the high quality, precise, secondary GI can be saved.
1.8. ***IM+QMC*** - slow, good precision, problem with blotches, primary GI can be saved.
1.9. ***IM+LC*** - the fastest, good precision, very few artifacts, quick setup and primary and secondary GI are saved.
1.10. ***IM+PM*** - quite fast, possible artifacts in corners, saves both primary and secondary.

2. EXTERIOR ANIMATION

- Objects and moving lights with changes in the color and geometry (DYNAMIC ANIMATION):

2.1. ***QMC+QMC*** - slow but acceptable, the most precise, excellent quality, general noise but easily removable.
2.2. ***QMC + LC*** - decent speed, precise, removable noise..
2.3. ***IM+QMC*** - fast, decent precision, not many artifacts. Flickering.
2.4. ***IM+LC*** - fast, quite precise, very few artifacts, primary and secondary GI savable, flickering.

- With only the camera in movement (STATIC FLY-THROUGH ANIMATION):

2.5. ***QMC+QMC*** - NOT RECOMMENDED, the slowest, noise present but removable, GI can not be saved.
2.6. ***QMC+LC*** - reasonably fast, very good quality, not many artifacts, secondary GI can be saved.
2.7. ***IM+QMC*** - fast, decent precision, not many artifacts, one can save and re-use primary GI.
2.8. ***IM+LC*** - fast and quite precise, not many artifacts, primary and secondary GI can be saved.

3. STATIC RENDERING

- Exterior:

3.1. *QMC+QMC* - slowest but acceptable, the most precise, high noise, GI cannot be saved.
3.2. *QMC+LC* - reasonably fast considering the high quality, precise, secondary GI can be saved.
3.3. *IM+QMC* - slow, good precision, noise in blotches, primary GI can be saved.
3.4. *IM+LC* - the fastest, good precision, very few artifacts, quick setup. Primary and secondary GI can be saved.
3.5. *IM+PM* - quite fast, possible artifacts in corners, saves both primary and secondary GI.

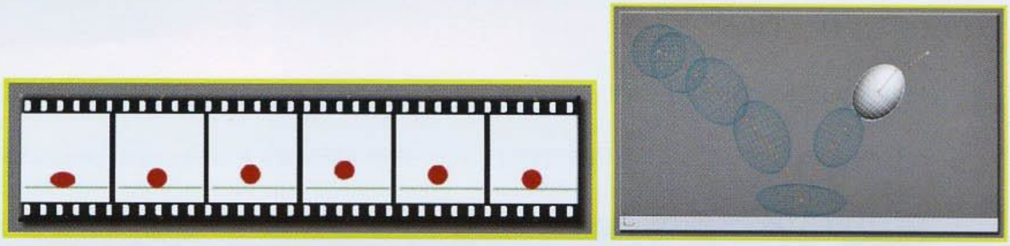
- Interior:

3.6. *QMC+QMC* - NOT RECOMMENDED, the slowest, noise present but removable, GI can not be saved.
3.7. *QMC+LC* - reasonably fast, very good quality, not many artifacts, secondary GI can be saved.
3.8. *IM+QMC* - quite fast with decent precision, not many artifacts, one can save and re-use primary GI.
3.9. *IM+LC* - fast and sufficiently precise, not many artifacts, primary and secondary GI can be saved.

These are the most common pairs of techniques used to calculate GI. Nothing stops one from using the other combinations, but from experience they can be defined as the most correct ones. Shortly, each method will be analyzed in-depth and the reasons for its use explained. The next step is to understand the concepts of static and dynamic animation.

ANIMATIONS

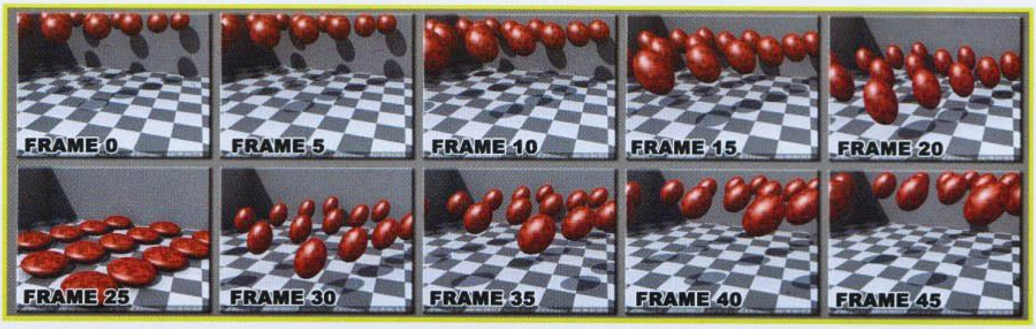
An animation is by definition a series of rapid sequential images. These are single renders, frames, arranged one after the other in such a way that if they are viewed in rapid succession, they create the illusion of fluid movement in the viewer.



■ **Figure 3.111**
Classic animation scene: a bouncing ball.

. DYNAMIC ANIMATION

The animation of a ball bouncing, rolling, etc... generates situations whereby the position of shadows changes, as well as other factors such as light color. When the ball is at its highest position it casts a shadow on the floor which, a few seconds later, changes shape and size because of it nearing the ground.



■ **Figure 3.112**
Some frames in an animation. In each instant the position, the light and the shadows change continuously. GI is different for each frame.

As well as the ball's position and its shadow, also the light will be subject to changes. It can happen that an area of the pavement is illuminated by the light of the sun, and in the following moment it can be completely in shadow. This type of animation is defined as dynamic.

. STATIC ANIMATION

Now imagine the same scene, with the ball having just hit the floor. Imagine freezing the scene. Time stops, everything is blocked, but it is possible to wander around and explore the "blocked" scenario. The only movement which is possible is one's point of view, nothing else moves. The shadow of the ball and likewise its position will remain in the same place, no matter where one decides to look at it from. This is a static animation, also called ***Fly-through***.
In this other example, one is walking through an apartment and walking around the rooms. Nothing moves. The doors are all open. The position of the sun remains the same and none of the light sources change appearance in any way. The only movement is the position of the viewer. Gradually, as the camera moves along its path, the observer gathers information about the shadows, light sources and the colors of geometric shapes. If we were to walk the same path backwards, we would realize that we already have this information, which has been stored previously. There is no need to calculate the GI again, because in the first "passing through" all the illuminotechnical information has been saved and the data can be recalled and reused.

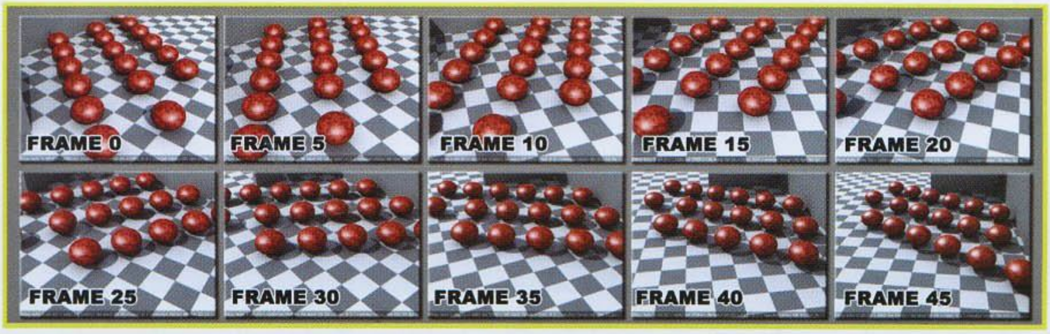


Figure 3.113
Frames in a static animation. Objects and shadows are always in the same position, no matter what angle the observer is at.

. STATIC RENDERING

Static rendering, images of single views, are nothing more than frames of static or dynamic animations. Once the concept of animation has been grasped, it is easily intuitable to think that an animation is the sum of a series of sequential static renders, placed in the right order and at a speed which gives one the idea of movement, and thus a film. So for static rendering, the same observations that we can make on animations are true. The only difference we can make is between static internal renders and static external ones.

After this panoramic view of Global Illumination calculation methods and the types of scenes, it is now time to analyze each of the four single methods of GI calculation and study the characteristics, and the pros and cons of each one. It will be much easier, this way, the reasons behind the previously explained associations between methods.

VRay: quasi-Monte Carlo (QMC)

INTRODUCTION

The *VRay: Quasi-Monte Carlo GI* rollout is a section which appears in the Renderer Panel of *VRay*, only if *quasi-Monte Carlo* method has been selected in the *Primary bounces* pull-down menu.

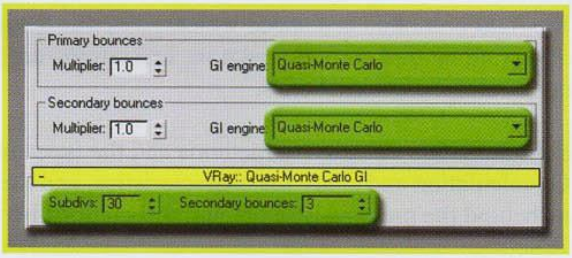
The *QMC* method calculates GI by means of a method we call "direct". GI is computed for each pixel, without any interpolation, independently from every other pixel. Its strong point lies in its extreme precision. However, this high level of precision requires a long rendering time.

Another negative aspect of *QMC* is the fact that it is impossible to save the GI, making it impossible to be re-used, for example in *Fly-through* animations. Even simple camera movements always, and in any case, require a new GI calculation.

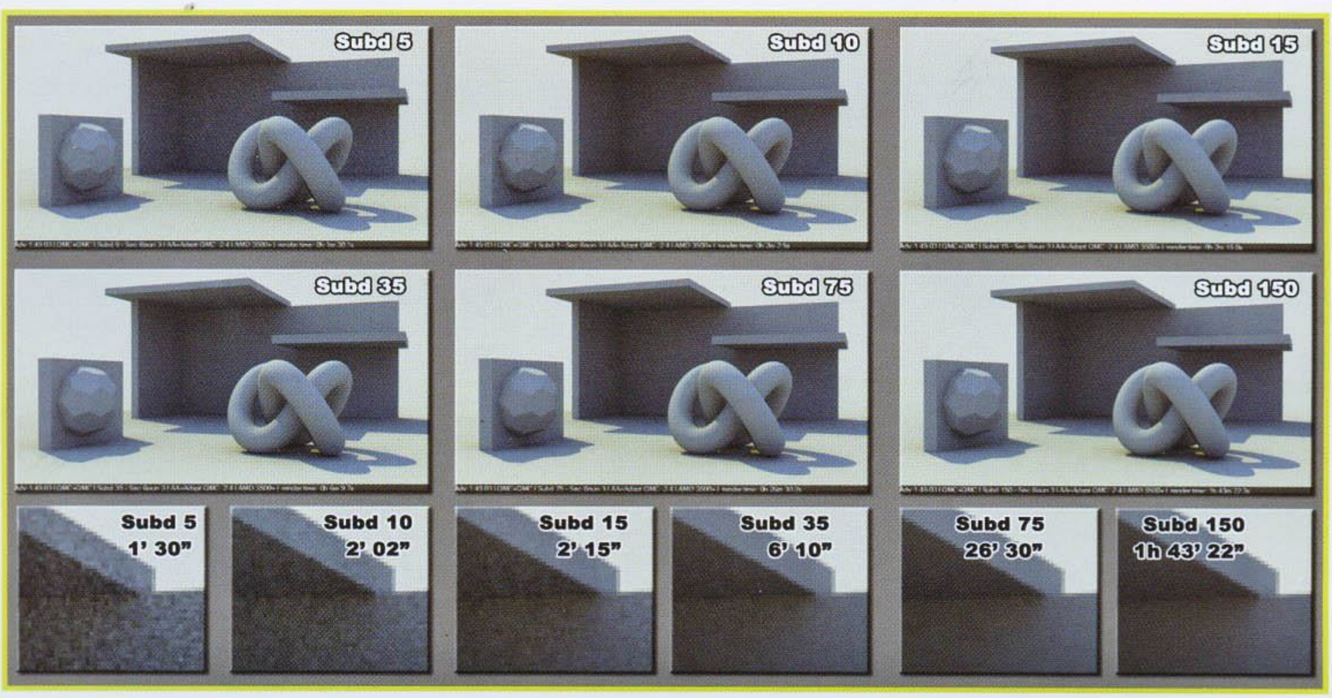
PARAMETERS

Subdivs - this parameter determines the number of rays used for GI calculation. Technically speaking, the real number of rays emitted by the point which is being elaborated is given by the square value of the number assigned to the *Subdivs* parameter. The greater the number of rays used, the better the rendering quality will be, thus reducing noise.

1	=	1	ray
2	=	4	rays
3	=	9	rays
8	=	60	rays
10	=	100	rays
100	=	10,000	rays



Some examples. Notice the increase in rendering time which follows the higher quality.



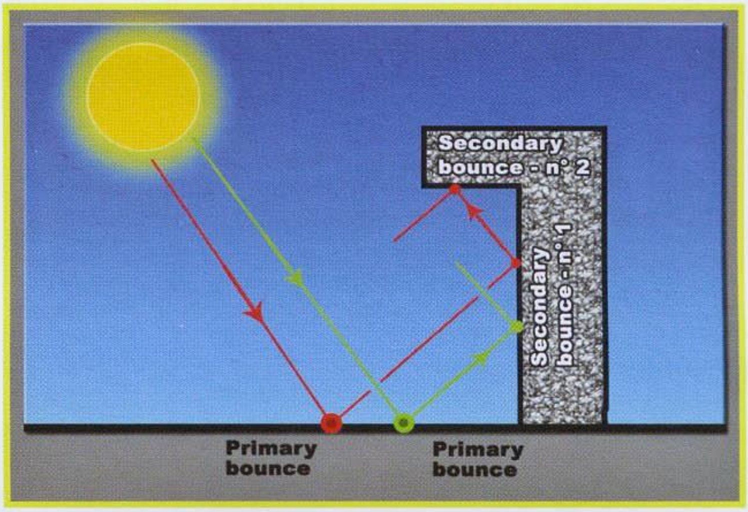
■ **Figure 3.114**
Chart showing the ***Subdivision*** parameter in the ***QMC*** method. This render was performed with a ***QMC+QMC*** method.

In an outdoor scene, as shown in the image, a ***QMC+QMC*** have been used for two reasons:

1. in order to demonstrate the noise reduction due to the increase in ***Subdivs***.
2. in order to demonstrate the actual possibilities of using ***QMC+QMC*** outdoors. The ***QMC*** method calculates GI for every single pixel in the image. At an equal Subdiv level, the greater the number of points is, that is, the resolution of the rendering, the greater the time of rendering will be. Vice versa, at an equal rate of resolution, the greater the number of subdivisions is, the cleaner the rendering will be. This will however increase the time it takes for rendering to be carried out. The ***QMC*** method, together with ***Progressive Path Tracing***, is one the most precise method in *VRay* and it is easy to see why it creates the best results.

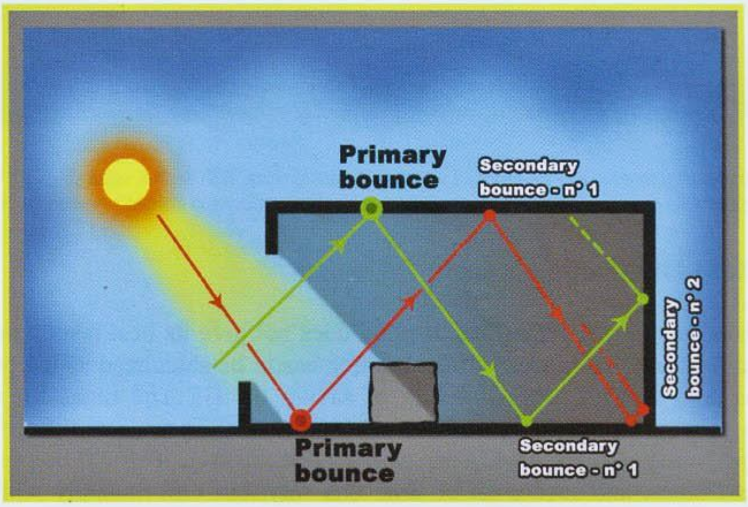
If possible (both for hardware reasons and because of the time it takes), it is advisable to use the ***QMC*** method always, especially for ***Primary bounces***, where GI calculation does not take long, because of the fact that only the ***Primary bounces*** are calculated. The ***Secondary bounces*** represent all the reflections after the primary ones, and require a great deal more data to be elaborated. For this reason, often one tends to use faster but less precise methods for ***Secondary bounces***.

In outdoor scenes, where ***Secondary bounces*** are not many, because of the nature of the scene itself, it is possible to use ***QMC*** also for the ***Secondary bounces***. In the scheme shown in the image, we can observe how two bounces only are enough to illuminate the scene, before the rays complete their cycle and are dispersed. By using the ***QMC*** method for ***Secondary bounces*** as well, although slow, one can obtain high quality results in a reasonable time.

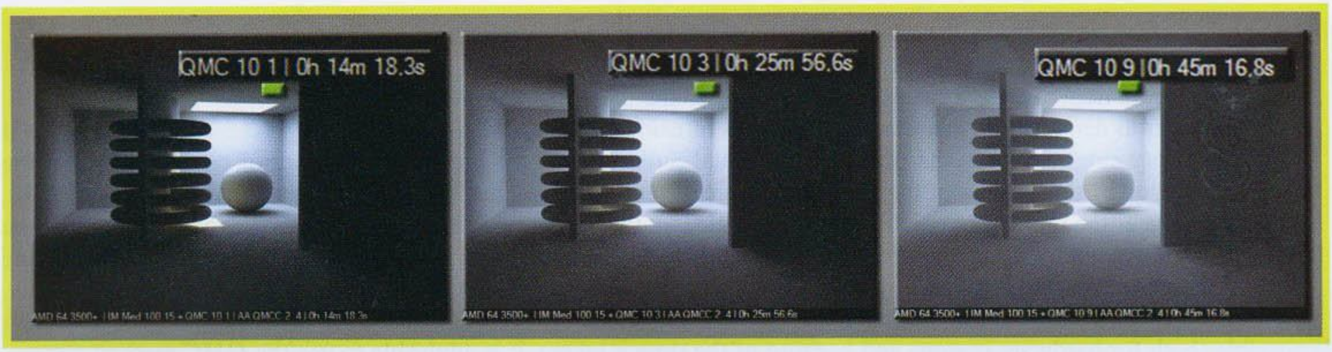


■ **Figure 3.115**
Scheme of an outdoor scene. The number of secondary bounces is extremely limited, suggesting the use of **QMC** also in the **Secondary bounces**.

Secondary bounces - this parameter sets the number of ***Secondary bounces*** of indirect light. The parameter ***Secondary bounces***, located in the ***VRay: quasi-Monte Carlo GI*** rollout, is active only if ***QMC*** has been selected in the ***Secondary bounces GI Engine*** pull-down menu. If ***QMC*** has been assigned only as the ***Primary Bounce GI Engine***, the ***Secondary bounces*** parameter is deactivated, because in this case ***QMC*** is being used only as an engine for the ***Primary bounces***. Thus it would be useless to indicate a parameter concerning ***Secondary bounces***, as it is no longer assigned to this method.



■ **Figure 3.116**
Scheme of an indoor scene. In this situation the number of Secondary Bounces is of fundamental importance for the quality of the render. Because of the geometry of the scene, light bounces many times. Technically VRay permits one to use the **QMC** method for secondary bounces, but in such a way the number of calculations required would be so great that it is preferable to use approximative methods such as **Light Cache** or **Photon Map**, allowing the reduction in calculation time.



■ **Figure 3.117**
Rendering of a scene with different values in **Secondary Bounces**: 1,3 and 9. As one can see, the more bounces are applied, the brighter the image becomes, because light can bounce to a greater depth. All this means a longer rendering time. The method used in this scene is **IM+QMC**.

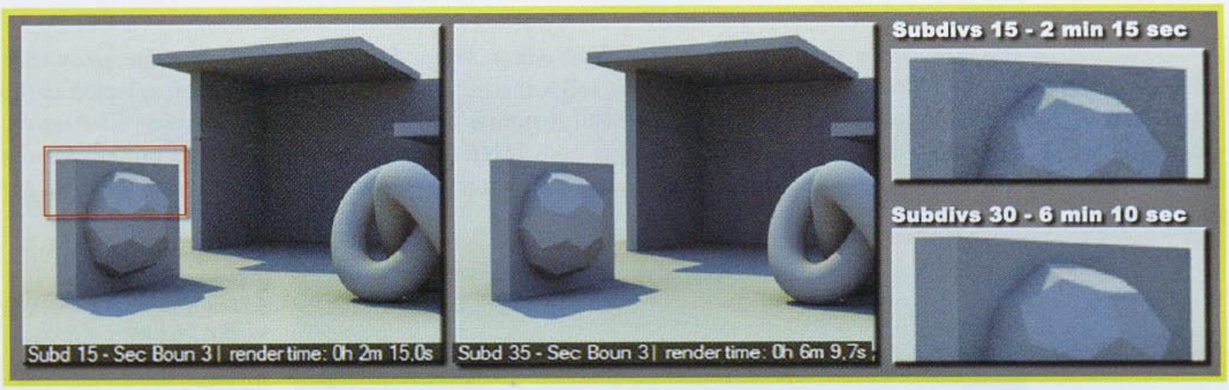
. INDOOR STATIC RENDERS

For internal rendering it is possible to pair ***QMC+QMC***, but it is time-consuming. The quality is excellent, extremely high, but in order to be able to eliminate noise many ***Subdivs*** are necessary. One of the advantages of this pairing is the setup, which is fast. The fact that there are only two parameters allows even beginners to obtain a quick setup.

■ **Figure 3.118**
QMC+QMC method. The image is very detailed but also noisy, as can be observed in the photo enlargement. Calculation time is near 20 minutes.

. OUTDOOR STATIC RENDERS

The ***QMC+QMC*** pair can instead be a good choice for outdoor renders. The number of bounces is not so great and rendering times are acceptable.



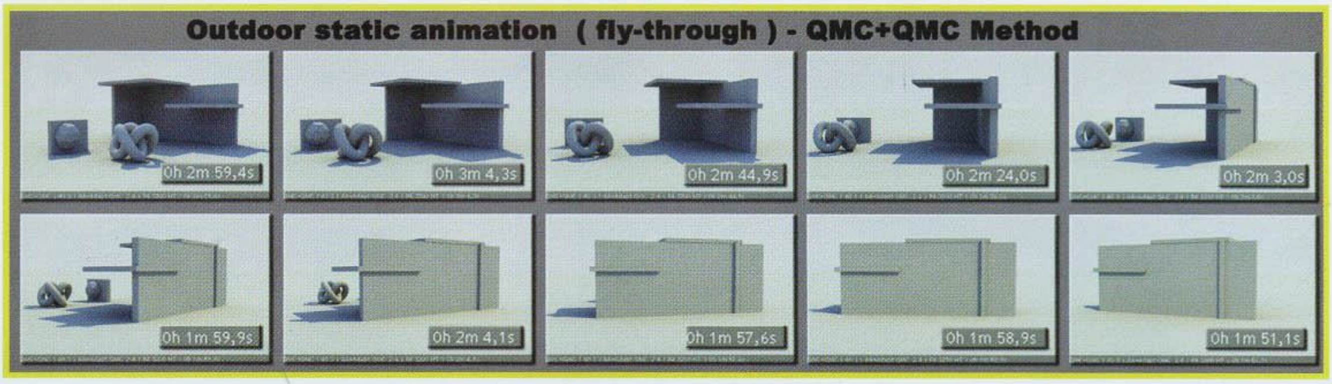
■ **Figure 3.119**
The **QMC+QMC** method. The image is very detailed and the shadows are clear-cut and precise. With **Sudivs** value set to 30, VRay provides a high quality render, with little noise and in a reasonable amount of time. The quality of GI is at its highest, thanks to the **QMC** method. 640x360 pixel resolution.

. INDOOR STATIC ANIMATIONS (Fly-Through)

It has just been demonstrated how the ***QMC+QMC*** method does not give its best results in closed environments. The situation worsens where animations are concerned. Each frame has to be calculated as if it were an independent render compared to the previous one, without any possibility of saving the GI. The times for rendering single frames is substantially the same as one would have for dynamic indoor animations. There is no difference because of the fact that one cannot save the GI and re-use it again. In ***Fly-through*** renders, in fact, it would be possible to re-use the GI data calculated in the previous frames, and save a lot of time this way, but unfortunately with the ***QMC*** method this is simply not possible.

. OUTDOOR STATIC ANIMATIONS (Fly-Through)

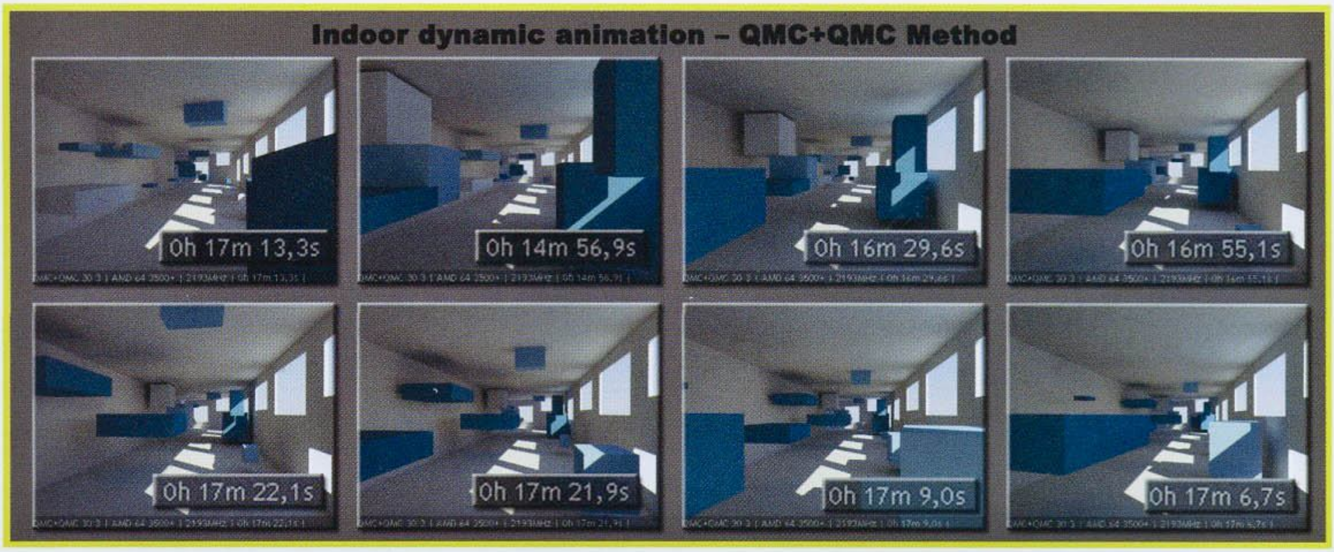
As seen in the previous case, the impossibility of saving the GI is a big problem. Although it is possible to use ***QMC+QMC*** efficiently in outdoor scenes, in the case of ***Fly-through*** animations it is best to find another solution. Considering the fact that objects are static, it is preferable to choose a method that allows one to save the GI , or at least the secondary one. It is, however, possible to use the ***QMC+QMC*** pair also for outdoor static animations, by means of high subdivisions and powerful hardware.



■ **Figure 3.120**
Some frames from an outdoor static animation. The scene is very simple and the resolution is of 500 x 300 pixels. With an average computer one can obtain clean images in less then 3 minutes per frame, with an excellent level of quality. Nothing moves except the observer's point of view.
--- DVD ANIMATION ---

. INDOOR DYNAMIC ANIMATIONS

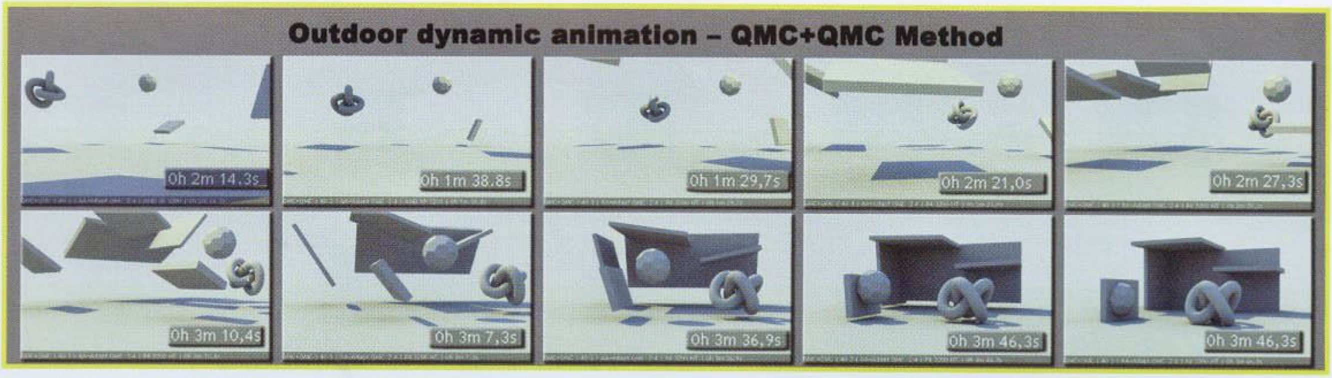
In dynamic animations it is necessary to calculate the GI for each and every frame. Hence, any save function of the GI is in vain. One might then think of using the ***QMC*** method for such animations. It is true that ***QMC*** can certainly be used in dynamic animations, but not as an engine for ***Secondary bounces***. The same observations which have been made regarding indoor static animation are applicable: the ***QMC+QMC*** method for indoor use, wether for renders or animations, is not recommended. In order to obtain a satisfactory level of quality, one must use many subdivisions and high ***Secondary bounce*** values, with a huge increase in rendering times. It is an unadvisable method, unless one has adequate hardware. In particularly complex and detailed scenes with numerous blur effects, it could be possible to think of using ***QMC+QMC*** for indoor dynamic animation.



■ **Figure 3.121**
Some frames in a dynamic internal animation. Because of the low value of ***Secondary bounces*** = 3, the luminosity of the scene is low. The grainy look of the animation is due to the low level of ***Subdivs***, which in any case, are set to 30. This value may be enough for outdoor scenes, but for indoor ones it is necessary to set this value higher. Already with these values the time per frame rendering is high, around 19 minutes, at a resolution of 400 x 300 pixels!
--- DVD ANIMATION ---

. OUTDOOR DYNAMIC ANIMATIONS

This is probably the only type of animation where the ***QMC+QMC*** method could realistically be used. In outdoor situations the ***QMC*** method is not that slow and provides excellent quality, providing one uses an adequate amount of subdivisions. With dynamic animations, ***QMC*** is thus a valid choice amongst methods. If the scene contains many details, small geometries or blurred specular reflections, ***Motion blur*** and ***Area Shadows***, ***QMC*** can definitely be of help both in terms of quality and time consumption.

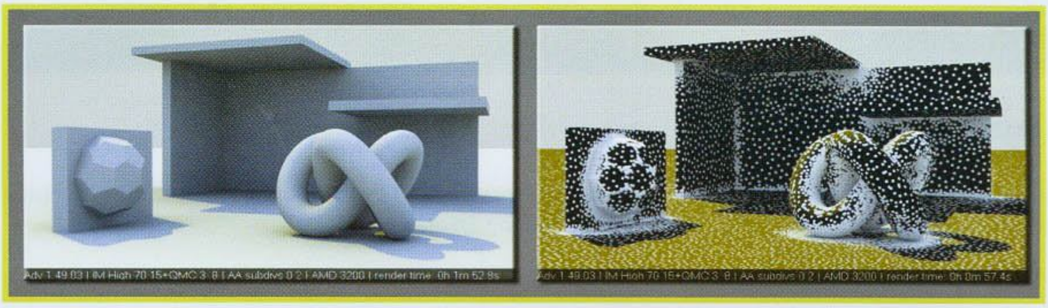


■ **Figure 3.122**
Some frames from an outdoor dynamic animation. The scene is very simple and the resolution is of 500 x 300 pixels. With an average computer it is possible to obtain clean images in less than 3 minutes per frame, with excellent quality.
- - - DVD ANIMATION - - -

VRay: Irradiance Map (IM)

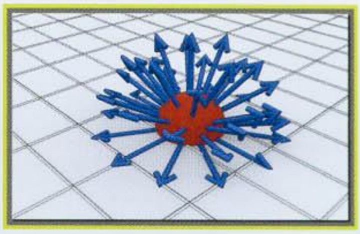
INTRODUCTION

Irradiance Map (***IM***) is one of the four methods for calculating GI. Compared to ***QMC***, ***IM*** is defined as approximative, because it is able to calculate GI only where there is most need, leaving out zones which are geometrically or materially simpler and do not require as many samples. It is also defined as View dependent, just like the ***QMC*** method.



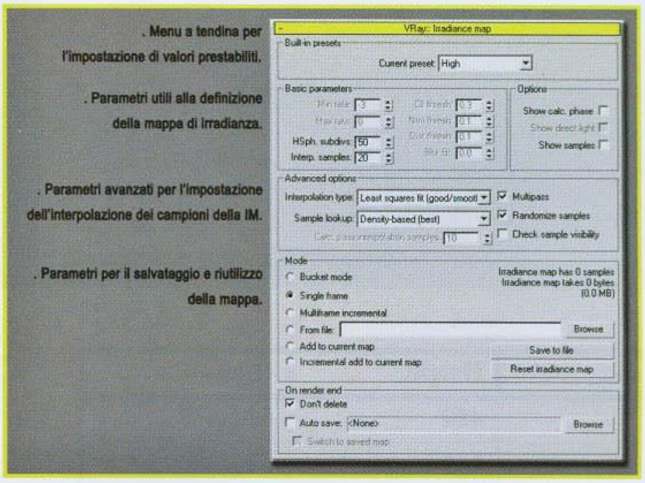
■ **Figura 3.123**
On the left, an image rendered with ***IM***. On the right, the representation of the samples used by VRay for the calculation of the GI. One can see that in particularly difficult areas, such as corners or curved surfaces, the number of samples is greater than flat surfaces or ones with few details.

In the right-hand image the ***Irradiance Map*** is shown. The ***IM*** is a three-dimensional map formed by points, visible in the image, which can be saved. Each point or sample contains information concerning the GI, whereas in the black areas the GI is calculated via interpolation between the closest points. These points are arranged in a strategic manner, in geometrically complicated areas, with strong color contrasts or differences in luminosity. Once their position has been decided, each point emits a certain amount of rays, with the precise aim of calculating the correct value of the GI for that sample.



■ **Figura 3.124**
A sample and the rays emitted for the calculation of the GI.

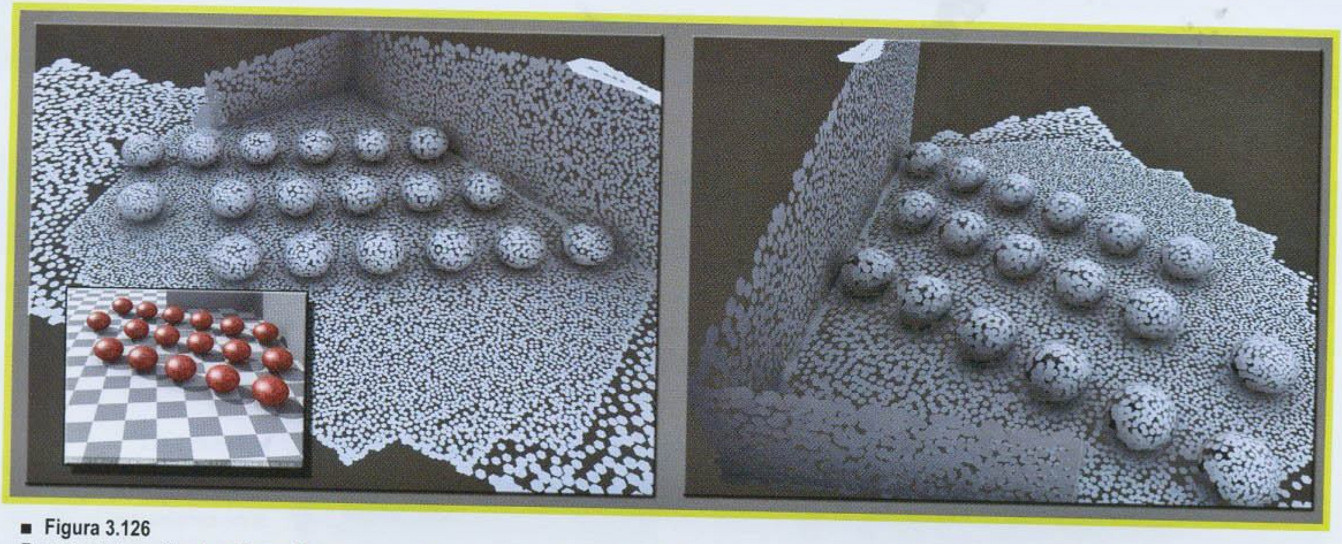
The difference compared to the ***QMC*** method is obvious. With ***QMC***, all pixels of the final render are processed for GI calculation, instead in the ***IM*** only some will be considered. The gathered data can be saved in an appropriate file, allowing it to be stored in case it is needed.



■ **Figura 3.125**
Rollout of the ***VRay: Irradiance map*** panel. The parameters for GI management are definitely many more than the two only ones for the ***QMC*** method.

The parameters in the ***VRay: Irradiance map*** rollout are very numerous compared to those for ***QMC***! Their purpose is to provide maximum flexibility in order to obtain the highest quality in the lowest possible time. Many of these parameters are strictly connected with each other, and it is not enough to simply use high values to obtain quality GI.

For this reason they will be examined one by one in the next few chapters, in order to understand their functions.



■ **Figura 3.126**
Representation of the ***Irradiance Map*** through the proprietary 3D visualizer, ***Irradiance map viewer***.

Before moving on to an in-depth analysis of each parameter, it is necessary first to have understood a rather complex concept: ***Prepass***. This term indicates the main processes which generate the ***Irradiance Map***.

PARAMETERS

Imagine having to render an image at a 640 x 480 pixel resolution. Divide the resolution by 2, then by 2 again and then lastly by 2 once more. The final size of the render will be 80 x 60 pixels.

	Width	Height	Operation
1st Subdivision	640	480	640x480
2nd Subdivision	320	240	640x480 /2
3rd Subdivision	160	120	640x480 /2/2
4th Subdivision	80	60	640x480 /2/2/2

■ **Figura 3.127**
Image rendered at resolutions which are multiples of 2.

Technically speaking, each time the resolution of a render is halved, the time of rendering is reduced by four times. An image of 320 x 240 pixels is 4 times smaller than an image of 640 x 480 pixels, whereas a render of 160 x 120 pixels is 4 times greater than one of 80 x 60 pixels. The concept of resolution reduction is used by *VRay* for calculating GI through the ***IM*** and the parameters which control this "subdivision" are ***Min rate*** and ***Max rate***. *VRay* calculates the ***IM*** through "passes". In the examined scene, the first one is carried out with an 80 x 60 resolution. It is very fast work, and GI calculation is quick when rendering an image at such a low resolution. During this ***Prepass***, *VRay* starts off "getting to know" the scene and its geometries. It gathers information and locates the more complex and detailed areas and the simpler ones. Once the first ***Prepass*** has been done, *VRay* begins with the second one, this time with a resolution twice the first one. Thanks to the information gathered in the first ***Prepass***, in the second one *VRay* can dedicate more time towards the more detailed areas, leaving out the areas with few details, such as flat and smooth surfaces. This way it is possible to save a lot of time. The rendering process via ***Prepasses*** continues until it reaches the final resolution fixed by the operator. Now the difference between ***QMC*** and ***IM*** is obvious. With ***QMC***, GI is calculated systematically, without considering geometric detail, whereas the ***IM*** uses a selective procedure. It is worth remembering, also, that the data of the samples obtained during the four ***Prepass*** procedures can be saved in the ***Irradiance Map***.

■ Figura 3.128
In the 80 x 60 pixel render, the samples, which are represented by white squares, are uniformly distributed over the whole image. As the resolution increases, VRay selects the critical areas, establishing more and more samples. In the image on the far right, one can see how the samples are concentrated at the base of the object and around the corners of the model, whereas in the flat areas samples are less dense.

How does one set the number of ***Prepasses*** to be applied? With the ***Min rate*** and ***Max rate*** parameters.

Basic parameters

Min rate - this parameter fixes the minimum resolution of the first ***Prepass***. ***Min rate*** = 0 means that the initial resolution to be calculated is the same as that of the rendering output. This way the ***IM*** would be similar to ***QMC***. With value set to -1, the resolution of the first ***Prepass*** is half that of the output. If it were set to -2, instead, the resolution of the first ***Prepass*** is a quarter of the resolution of the output. In 99.9% of cases ***Min rate*** is < 0.

Max rate - this parameter fixes the maximum resolution of the last ***Prepass*** in ***IM*** calculation. With detailed models, it is necessary to use very high levels, which in extreme cases coincide with ***Max rate*** = 0.

Thus we have two parameters which indicate what minimum resolution to start calculating the ***IM*** (***Min rate***) and at what maximum resolution to stop (***Max rate***). In order to better understand this particular concept, a practical example is provided, using an image rendered at 640 x 480 pixels. Let us begin with the following values:

Min rate = -3 **Max rate = 0**

In order to discover the minimum starting resolution for ***IM*** calculation, one must read the value of ***Min rate***. In this case it is -3. One must then divide the resolution of 640 x 480 pixels firstly by 2, then by 2, then once more by 2 again, in other words 8 times. The resolution of first ***Prepass*** is 80 x 60 pixels:

640	x 480	/ 2	=	320	x	240	**3rd PREPASS**	at 320 x 240 pixels
320	x 240	/ 2	=	160	x	120	**2nd PREPASS**	at 160 x 120 pixels
160	x 120	/ 2	=	80	x	60	**1st PREPASS**	at 80 x 60 pixels

-3 times More precisely 2^3=8, divide by 8 the output resolution, (640 x 480 pixels) and one has the initial resolution of the 1st ***Prepass*** (80 x 60 pixels).

At this point, as a last unknown factor, one must discover how to set the resolution of the last ***Prepass***. In the previous example one can observe how *VRay* reaches and calculates 4 ***Prepasses***. The management of this operation is assigned to the ***Max rate*** parameter.

In the current exercise ***Max rate*** has been set to 0. This means that the last ***Prepass*** will be calculated at output resolution:

Min rate **= -3**
Max rate **= 0**

2^0 = 1 so….

640	x 480	/ 1	=	640	x	480	**4th PREPASS**	at 640 x 480 pixels

In short, with *VRay* values set to:

Min rate **= -3**
Max rate **= 0**

4 ***Prepass*** procedures will take place:

640	x 480	/ 1	=	640	x	480	**4th PREPASS**	at 640 x 480 pixels
640	x 480	/ 2	=	320	x	240	**3rd PREPASS**	at 320 x 240 pixels
320	x 240	/ 2	=	160	x	120	**2nd PREPASS**	at 160 x 120 pixels
160	x 120	/ 2	=	80	x	60	**1st PREPASS**	at 80 x 60 pixels

One last but important consideration. The ***IM***, as well as depending on viewing, also depends on the final output resolution. This means that correct values of ***Min rate*** and ***Max rate*** for a 640 x 480 pixel resolution, may not be correct for a resolution of 1280 x 960 pixels, for instance. Or rather, if one wanted to obtain the same ***IM*** quality as a 640 x 480 resolution, but for an output resolution of twice as much, the ***Min rate*** and ***Max rate*** parameters would have to be modified. The calculation is not difficult.

Now the aim of this exercise is to render an image at a final resolution of 1280 x 960 pixels, obtaining the same quality as was obtained at a resolution of 640 x 480 pixels.

In the previous example, the first ***Prepass*** was calculated at 80 x 60 pixels, whereas the last was at 640 x 480 pixels. So one must obtain a ***Prepass*** of the same resolution. This way, the ***IM*** will be identical, although the final output will not be. The following values are to be used:

Min rate **= -4**
Max rate **= -1**

1280	x 960	/ 2	=	640	x	480	**4th PREPASS**	a 640 x 480 pixels
640	x 480	/ 2	=	320	x	240	**3rd PREPASS**	a 320 x 240 pixels
320	x 240	/ 2	=	160	x	120	**2nd PREPASS**	a 160 x 120 pixels
160	x 120	/ 2	=	80	x	60	**1st PREPASS**	a 80 x 60 pixels

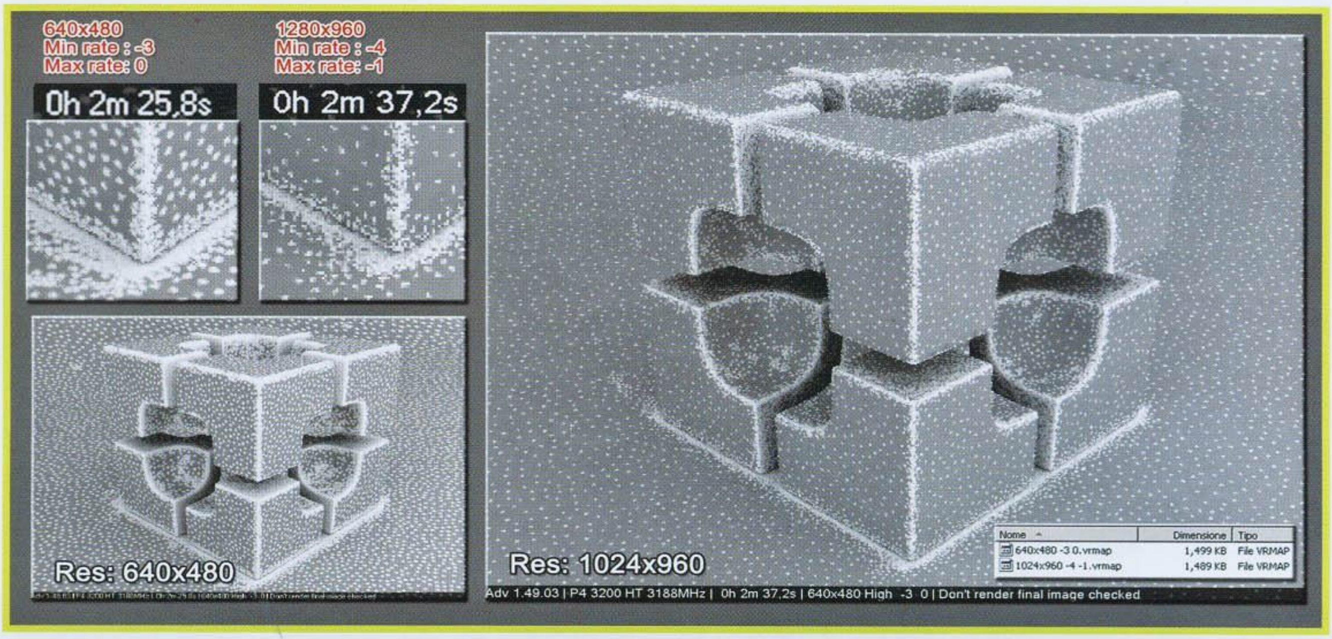
By setting the value ***Min rate*** = -4, the initial resolution for ***IM*** calculation is equal to 2^4=16, in other words:

1280/16 = 80 pixels
1024/16 = 60 pixels

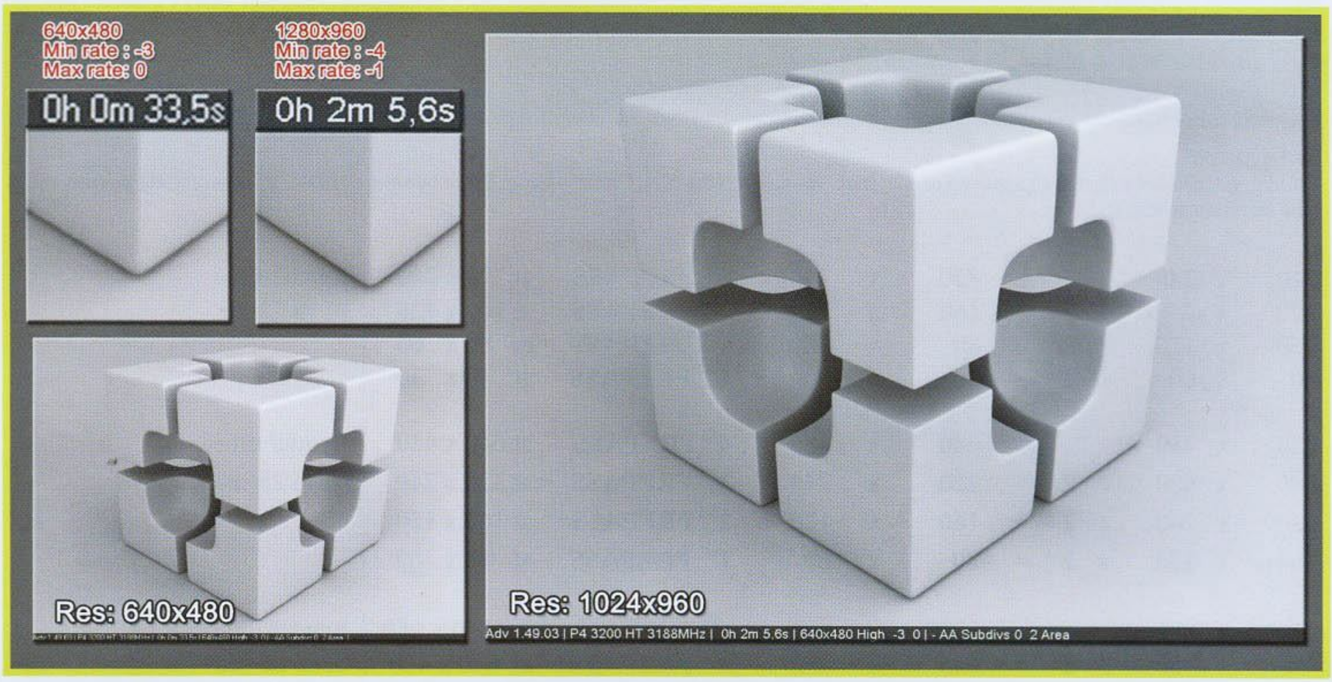
By setting the value ***Max rate*** = -1, the resolution for the calculation of the 4th ***Prepass*** is equal to 2^1=2, in other words:

1280/2 = 640 pixels
1024/2 = 480 pixels

It is possible to obtain the same result by following a different path. It consists of always rendering the image at 640 x 480 pixels and with values ***Min rate*** = -3 and ***Max rate*** = 0; at this point the ***IM*** will be saved. One can then load this file and re-launch the render at a resolution of 1280 x 960 pixels using the previously calculated ***IM***. The result will be identical. The potential and extreme flexibility of the ***IM*** is easily understandable.



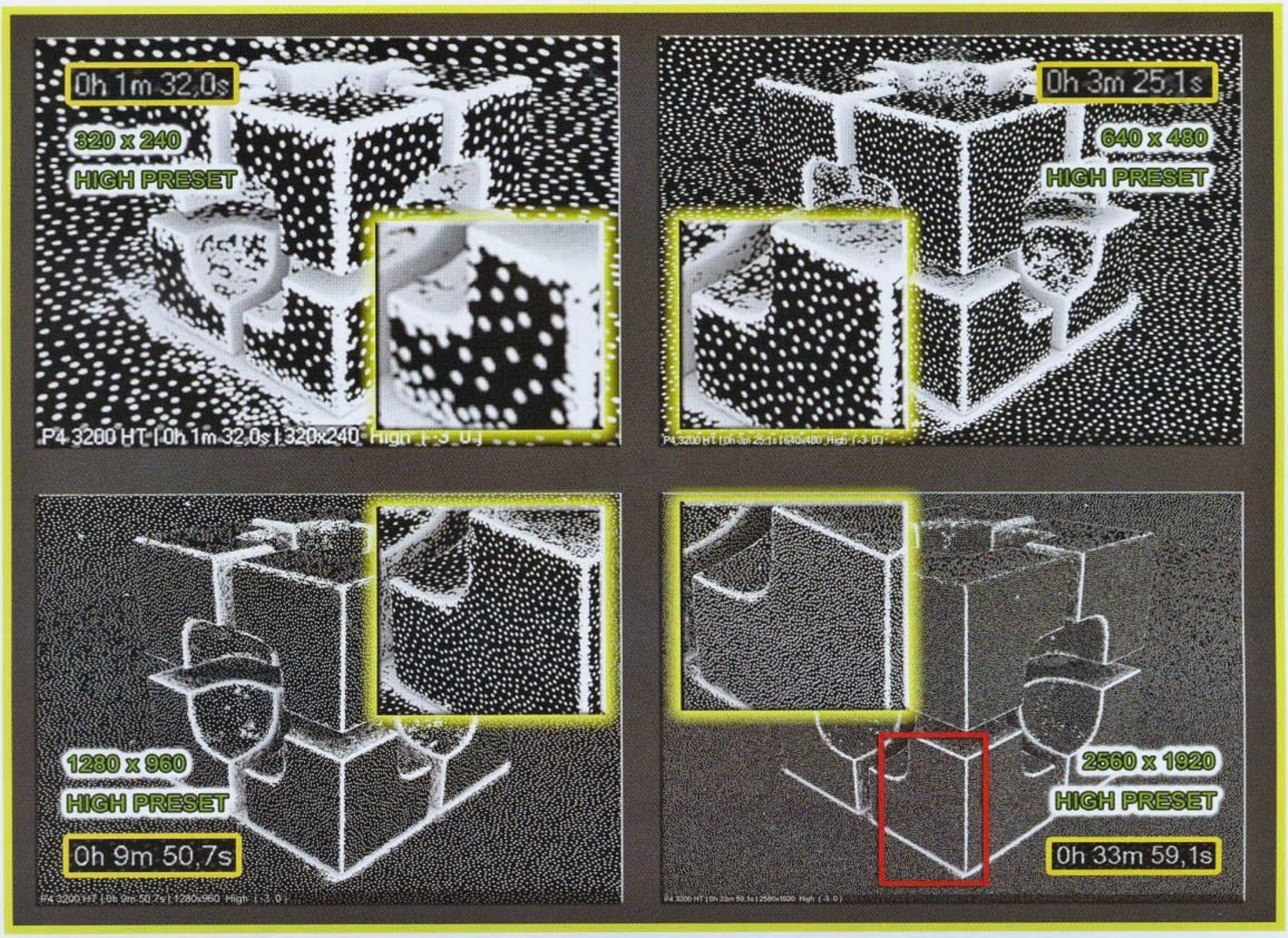
■ **Figura 3.129**
One can see that the quantity and distribution of the samples is almost identical, even at different resolutions. One must not be mislead by the size of the samples. In the image at 640 x 480 pixels, the samples appear to be twice as big as the ones in the image at 1280 x 960 pixels. This is because they depend on the size of the output, just like the ***IM***. There is yet another confirmation regarding the equality of the GI solution: first of all, the time for GI calculation, which is exactly the same, even at different resolutions, as well as the size of the file containing the ***Irradiance Map***.



■ **Figura 3.130**
Instead in the final rendering process, the duration time is different. This is due to the fact that calculating an image of twice the resolution, takes around 4 times the time required for an image of half the size. The GI quality is the same though.

In the ***VRay: Irradiance map*** rollout there is also a pull-down menu containing ***presets***. These have been created by Chaosgroup, so that some parameters can be automatically setup, amongst which ***Min rate*** and ***Max rate***, for renders in low, medium and high quality at a known resolution of 640 x 480 pixels. As a result, for resolutions which differ from 640 x 480 pixels, ***Min rate*** and ***Max rate*** will have to be changed manually. If not, a 3000 x 4000 pixels render, set at preset High, ***Min rate*** = -3 and ***Min rate*** = 0, would have incredibly long rendering times. This is because the first ***Prepass*** of the ***IM*** would be calculated at a very high resolution, thus obtaining a great number of samples in areas without much detail.

To confirm this phenomenon, observe the next example, where ***preset High*** is applied and where some images will be rendered at resolutions which are multiples of 2, starting from 320 x 240 pixels, and reaching 2560 x 1920 pixels.

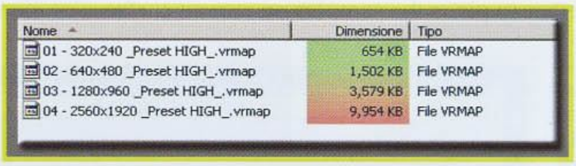


■ **Figura 3.131**
Renders with different solutions, always using ***preset: High***. The quality of the ***IM***, at the same values of ***Min rate*** and ***Max rate***, changes radically, as does also the rendering time.

320	**x**	**240**		=	320	x	240	**4th PREPASS**	at 320 x 240	Render time
320	x	240	/ 2	=	160	x	120	**3rd PREPASS**	at 160 x 120	01 m 32 sec
160	x	120	/ 2	=	80	x	60	**2nd PREPASS**	at 80 x 60	
80	x	60	/ 2	=	40	x	30	**1st PREPASS**	at 40 x 30	
640	**x**	**480**		=	640	x	480	**4th PREPASS**	at 640 x 480	Render time
640	x	480	/ 2	=	320	x	240	**3rd PREPASS**	at 320 x 240	03 m 25 sec
320	x	240	/ 2	=	160	x	120	**2nd PREPASS**	at 160 x 120	
160	x	120	/ 2	=	80	x	60	**1st PREPASS**	at 80 x 60	
1280	**x**	**960**		=	1280	x	690	**4th PREPASS**	at 1280 x 960	Render time
1280	x	960	/ 2	=	640	x	480	**3rd PREPASS**	at 640 x 480	09 m 50 sec
640	x	480	/ 2	=	320	x	240	**2nd PREPASS**	at 320 x 240	
320	x	240	/ 2	=	160	x	120	**1st PREPASS**	at 160 x 120	
2560	**x**	**1920**		=	2560	x	1920	**4th PREPASS**	at 2560x1920	Render time
2560	x	1920	/ 2	=	1280	x	960	**3rd PREPASS**	at 1280 x 960	33 m 59 sec
1280	x	960	/ 2	=	640	x	480	**2nd PREPASS**	at 640 x 480	
640	x	480	/ 2	=	320	x	240	**1st PREPASS**	at 320 x 240	

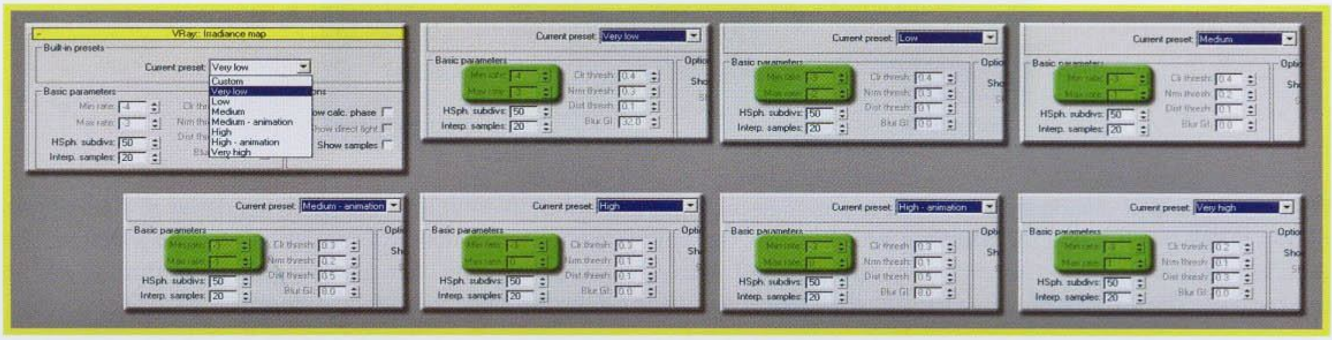
By using fixed ***Min rate*** and ***Max rate*** values, and changing output resolution, rendering time increases considerably, and so does the number of samples generated. The higher the resolution is pushed, the more samples are computed and used. This is certainly an advantage for GI quality. The more samples are produced, the greater the precision and quality of the ***IM*** will be. Bear in mind, though, that ***preset: High*** values are set for a 640 x 480 resolution. One can then understand that if the same values of ***Min rate*** and ***Max rate*** are used for a 2560x1920 pixels resolution, these values can then be defined as a ***really Super High preset***!

To confirm the fact that with fixed ***Min rate*** and ***Max rate*** values, changing only the resolution, the ***IM*** quality changes, it is enough to look at the size of the file ***.vrmap*** in MBytes.



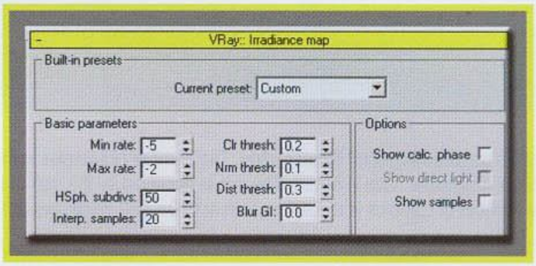
■ **Figura 3.132**
The increase in size of the file ***.vrmap***, the save file for the ***IM***, due to doubling output resolution with ***Min rate*** = 0 and ***Max rate*** = - 3.

During the explanation of the ***Min rate*** and ***Max rate*** parameters, the ***preset*** pull-down menu was mentioned.



■ **Figura 3.133**
The ***presets*** are predefined settings created by the authors of VRay to help setting up scenes.

Inside the menu, some "factory-made" values are to be found, which allow one to automatically setup ***IM*** values for low, medium or high quality at a resolution of 640 x 480 pixels. Looking carefully, one notices that not only ***Min rate*** and ***Max rate*** values change, but also parallel settings. It is to be remembered that only ***Max rate*** and ***Max rate*** settings depend on resolution. The darkened settings which are non-modifiable grey-color values do not depend on resolution. This means that for a good quality render at 3500 x 2625 pixels, one usually uses the ***High preset***, which assures high values on all the parameters: ***Min rate*** and ***Max rate***, ***Clr thresh***, ***Nrm. thresh***, ***Dist. thresh*** and ***Blur GI***. Then one can modify the preset in the pull-down menu under ***Custom***. This operation leaves all the previous parameters unchanged in ***High*** modality, giving one the possibility to edit them. At this point, one can change only the ***Min rate*** and ***Max rate*** values, in order to set them to correct values (see preceding paragraph).

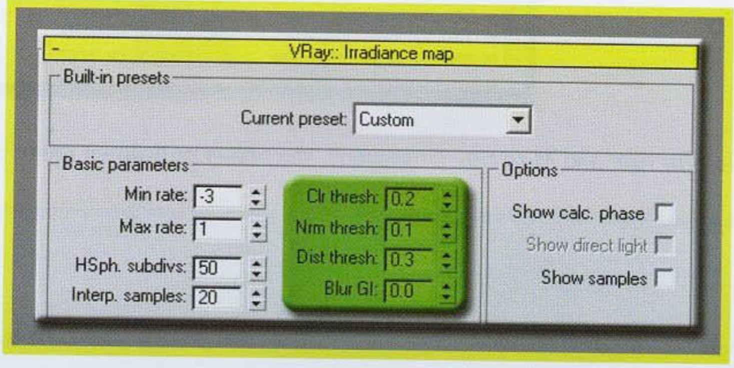


■ **Figura 3.134**
By setting the ***preset*** as ***Custom***, one can edit all the parameters of the ***VRay: Irradiance map*** rollout. The image shows settings for a good quality render at a resolution of 3500 x 2625 pixels.

Current preset - the ***presets*** in the pull-down menu are a list of preset parameters which allow one to modify the parameters of the ***Irradiance Map*** automatically. There are seven:

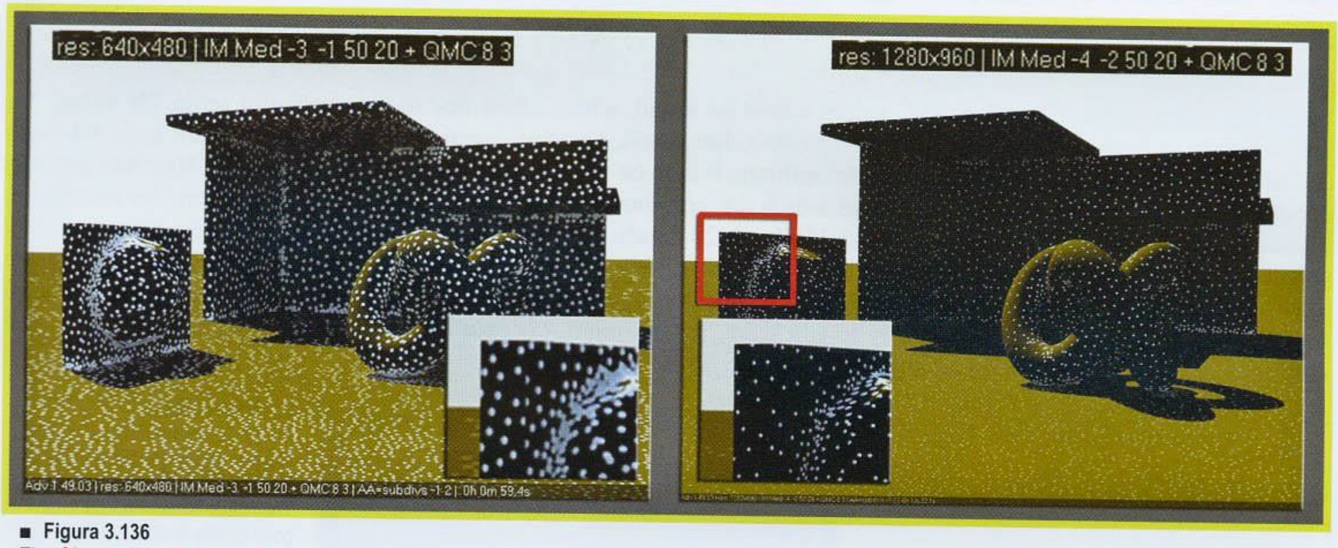
Very low	-	This ***preset*** generates very fast renders, which are useful only for pre-visualization.
Low	-	***Preset*** with low parameters. Quality is better than the ***Very Low preset***, but still for using in preview.
Medium	-	Can be used in many situations, except for in scenes with very small size objects.
Medium Animation	-	This ***preset***, useful during animations, works towards eliminating flickering problems during GI animations. The ***Distance threshold*** parameter, in fact, is very high.
High	-	***Preset*** with high values. It allows the calculation of high quality GI. This ***preset*** can be used both in scenes containing small objects and also in animations.
High Animation	-	This is a ***preset*** to be used in case the ***High preset*** is not sufficient to obtain high quality animations. In this case the ***Distance threshold*** parameter is very high.
Very High	-	To be used in scenes with an extremely high number of small objects and when one wants to obtain a high GI quality. Rendering times are very high.

With ***Min rate*** and ***Max rate*** parameters it is possible to control the number of ***Prepass*** processes, and consequently the ***IM*** quality. The remaining values are used for managing the number of and position of the samples. This makes it possible to establish wether to place the samples on curved surfaces or along the corners of a wall.



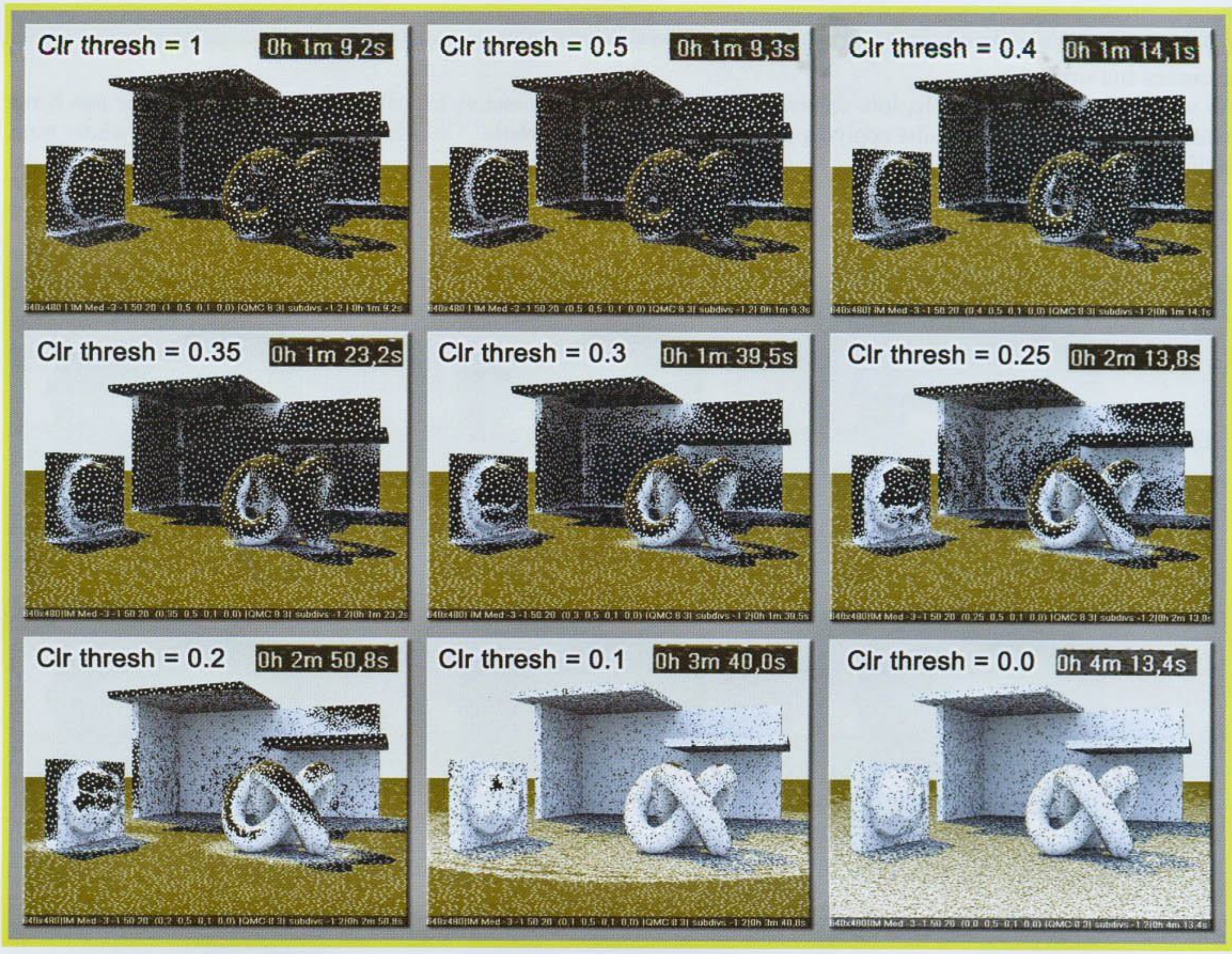
■ **Figura 3.135**
The values ***Color***, ***Normal*** and ***Distance threshold*** are parameters which are used to define the position of samples.

First of all we will demonstrate how these parameters do not depend on resolution. A scene is rendered at a resolution of 640 x 480 pixels using the ***Medium preset***. After, the same scene is elaborated at a resolution of 1280 x 960 pixels, exactly twice as much. In this case only ***Min rate*** and ***Max rate*** values are to be changed, bringing them from -3 and -1 to -4 and -2. The ***Clr thresh***, ***Nrm thresh***, ***Dist thresh*** and ***Blur GI*** are the same in both resolutions.



■ **Figura 3.136**
The ***Clr***, ***Nrm*** and ***Dist threshold*** values are not influenced by resolution. The number of samples is the same in both images. Visually, the samples seem to be more numerous in the render at 640 x 480 pixels. This is not so. The eye is deceived by the size of the sample itself which appears to be larger at low resolutions. If one looks carefully, the number of samples is the same, both at 640 x 480 pixels and at 1280 x 960 pixels.

Color threshold (Clr thresh) - this parameter controls the sensitivity of the ***IM*** algorithm according to the variation of indirect light. High values mean less sensitivity, low values mean more sensitivity, with consequential increase in the number of samples.
In the next example, the same scene is rendered at a resolution of 640 x 480 pixels with the ***Medium preset***, and this is done a few times varying the ***Clr thresh*** value. For this test the ***Nrm threshold*** parameter is set at 0.5. This way, sample disposition is completely managed by the ***Color threshold*** parameter.

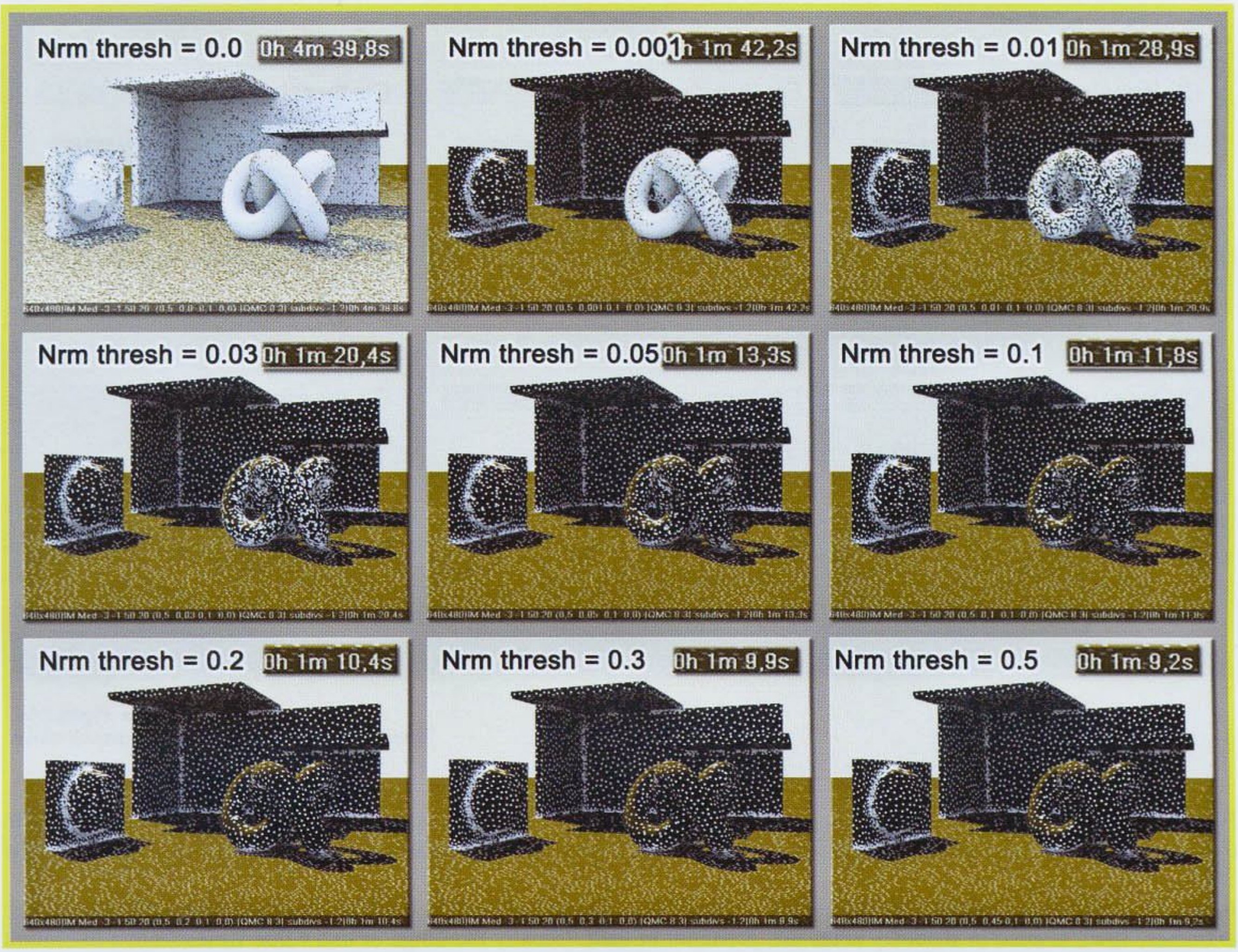


■ **Figura 3.137**
Rendered images with different ***Color threshold*** values.

By diminishing the ***Clr thresh*** parameter value, not only can one notice an increase in the number of samples. It is in the areas with most light contrast, in the areas where well-lit and shadowy areas meet, that ***Clr. thresh*** concentrates its efforts.

In the areas which are directly hit by sunlight, sample position and amount is more or less the same (unless values re extremely low), instead in shadowy areas this is cannot be said. Usually the range of this value varies from a minimum of 0,25, a very low value which generates excellent quality ***IM*** but long rendering times, to a maximum of 0.4, above which one does not notice any significant variation in the number or position of the samples.

Normal threshold (Nrm thresh) - this parameter controls the sensitivity of the ***IM*** algorithm according to the variation of surface normals and superficial details. High values mean less sensitivity, low values mean more sensitivity to curves and small details.
As we saw in the last example, low values will be used for all parameters except ***Nrm thresh***, so that one can better evaluate sample variation whilst applying changes to ***Normal threshold***. The ***Medium preset*** is used the whole time, taking ***Clr thresh*** from 0.4 to 0.5.



■ **Figura 3.138**
Images rendered with different values of ***Normal threshold***.

This value acts mainly in areas with strong differences in the normals of surfaces. Its action is quite evident on the curved surface of the geometry in the centre of the image. This parameter involves the whole model, regardless of what is exposed to direct light or not. It is very sensitive, and its effects, in this case, can be appreciated in the range between 0.001 and 0.1. Higher values do not greatly change the effect.

Distance threshold (dist thresh) - this parameter controls the sensitivity of the *IM* algorithm according to the distance between two or more surfaces. Value 0.0 means that the *IM* will not be influenced by the distance between the surfaces, whereas with high values, *VRay* will place more samples around polygons which are very near to each other. High values mean less sensitivity, low values mean more sensitivity to curves and small details.
As we saw in the last example, low values will be used for all parameters ecept ***Dist thresh***, so that one can better evaluate sample variation whilst applying changes to ***Distance threshold***. The ***Medium preset*** is used, taking ***Clr thresh*** and ***Nrm thresh*** from 0.4 to 0.5, thus eliminating their effect on the *IM* calculation.

■ **Figura 3.139**
Images rendered with different values of ***Distance threshold***.

The increase of the ***Distance threshold*** value gives, as a consequence, an increase in the number of samples in zones where two or more polygons are closest, regardless of the angle of the surface or the exposure to direct light. If this parameter is too high, it can cause long rendering times, without any improvement in the *IM* quality.

Blur GI - this parameter, useful in animations, blurs the GI before the *IM* is rendered. It allows the reduction of flickering, a problem which occurs in areas of small detail and during animations with moving objects. It is measured in pixels and usually used with a range from 1 to 10.

In the next few examples, all the parameters are used with the ***Medium preset***. As flickering problems usually occur in corners, this parameter creates a slight variation in the *IM* near these corners, mixing the samples of these areas. This is *VRay*'s method for eliminating flickering problems. The side-effect is a loss of detail in the GI (this parameter has been eliminated in version 1.5).

Distance threshold (dist thresh) - this parameter controls the sensitivity of the *IM* algorithm according to the distance between two or more surfaces. Value 0.0 means that the *IM* will not be influenced by the distance between the surfaces, whereas with high values, *VRay* will place more samples around polygons which are very near to each other. High values mean less sensitivity, low values mean more sensitivity to curves and small details.
As we saw in the last example, low values will be used for all parameters ecept ***Dist thresh***, so that one can better evaluate sample variation whilst applying changes to ***Distance threshold***. The ***Medium preset*** is used, taking ***Clr thresh*** and ***Nrm thresh*** from 0.4 to 0.5, thus eliminating their effect on the *IM* calculation.

■ **Figura 3.139**
Images rendered with different values of ***Distance threshold***.

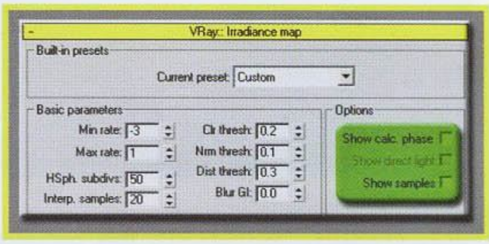
The increase of the ***Distance threshold*** value gives, as a consequence, an increase in the number of samples in zones where two or more polygons are closest, regardless of the angle of the surface or the exposure to direct light. If this parameter is too high, it can cause long rendering times, without any improvement in the *IM* quality.

Blur GI - this parameter, useful in animations, blurs the GI before the *IM* is rendered. It allows the reduction of flickering, a problem which occurs in areas of small detail and during animations with moving objects. It is measured in pixels and usually used with a range from 1 to 10.

In the next few examples, all the parameters are used with the ***Medium preset***. As flickering problems usually occur in corners, this parameter creates a slight variation in the *IM* near these corners, mixing the samples of these areas. This is *VRay*'s method for eliminating flickering problems. The side-effect is a loss of detail in the GI (this parameter has been eliminated in version 1.5).

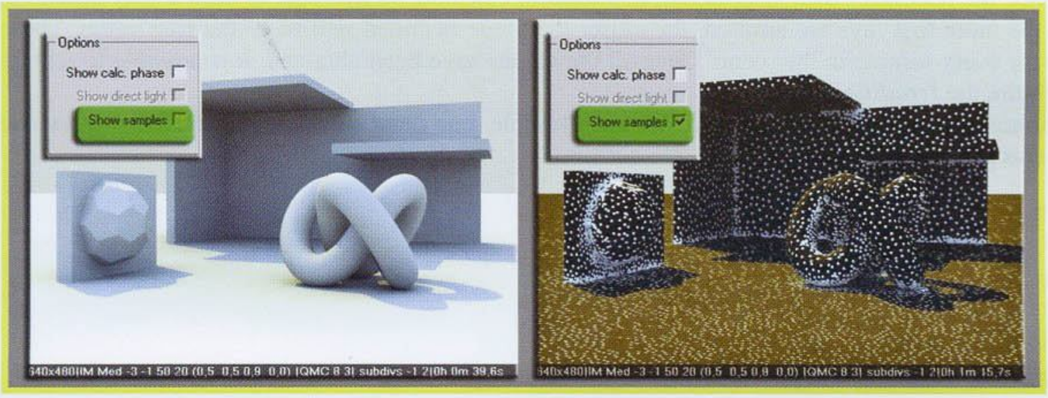
■ **Figura 3.140**
Images rendered with different ***Blur GI*** values. The first sequence has the ***Show Samples*** parameter activated. In the second one it has been deactivated.

On the right-hand side of the ***VRay: Irradiance map*** rollout, there is a small box called ***Options*** which contains three new options. These parameters are only for service, as they do not influence the actual ***IM*** calculation or the GI quality. For the exercises the ***Medium preset*** is used.



■ **Figura 3.141**
The parameters inside the ***Options*** box are useful for observing and better understanding what is happening during the ***IM*** calculation.

Show samples - if activated, at the end of the GI calculation *VRay* does not undergo the final render, but instead a series of points corresponding to the samples which would be used for ***IM*** calculation. Useful for studying the ***Irradiance map***.

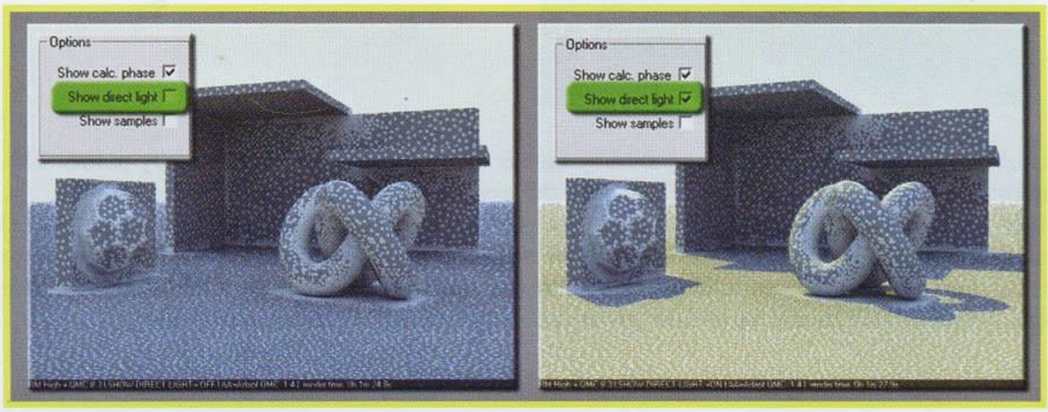


■ **Figura 3.142**
By activating the ***Show samples*** parameter, it is possible to see the ***IM*** samples "live".

Show calculation phase - with this option activated, *VRay* shows the ***IM*** calculation within the ***Frame buffer***. It is thus possible to have a general idea of how the scene will be illuminated, without having to wait for the whole ***IM*** calculation. So it is useful to keep it activated in order to have feedback about what is happening. It is possible that, with this parameter active, there may be a slight increase in rendering times, especially for images of great size.

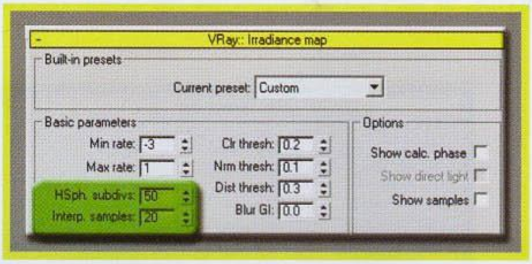
■ **Figura 3.143**
By setting ***Min rate*** = -3 and ***Max rate*** = 0, ***VRay*** generates four ***Prepasses***.

Show direct light - this option is available, only if ***Show calculation phase*** is active. It allows one to see, inside the ***Frame buffer***, the direct light during ***IM*** calculation. The aim of this option is to provide further visual information on what *VRay* is calculating.



■ **Figura 3.144**
By activating the ***Show direct light*** option, one can observe, during the ***IM*** calculation, the effect of direct light. This does not influence the render quality in any way.

The two final parameters are: ***HSph. subdivs*** and ***Interp. samples***.



■ **Figura 3.145**
The ***HSph. subdivs*** and ***Interp. Samples*** parameters.

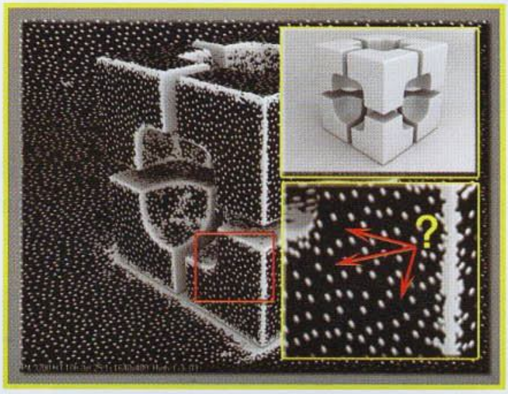
Hemispheric subdivision (HSph. Subdivs) - this parameter controls the number of ***Subdivisions*** of each single sample. Low values mean shorter rendering times, but also lower quality. Low values will also introduce dark blotches in the GI of the final result. High values, instead, produce better results, in exchange for time.

For each sample which is present in the scene, ***VRay*** must calculate its value: the GI. From each sample a certain amount of rays are emitted in a random manner, using the ***quasi-Monte Carlo*** algorithm. This way the area around the sample is analyzed and, once the necessary data is gathered, it is used to calculate an average, in such a way obtaining the GI value. Obviously, the more that rays are emitted, the closer the value obtained will be to the true value. These operations are carried out for every sample in the scene. Once all the values have been obtained, it is possible to reunite them all in one file and acquire the ***Irradiance Map***.
In order to obtain the total number of rays that ***VRay*** uses for each sample, one must calculate the square of the value inserted in the ***HSph. subdivs*** parameter:

. HSph. subdivs	:	1	=	1	ray
. HSph. subdivs	:	2	=	4	rays
. HSph. subdivs	:	10	=	100	rays
. HSph. subdivs	:	25	=	625	rays
. HSph. subdivs	:	50	=	2500	rays
. HSph. subdivs	:	100	=	10000	rays

In most cases, in order to obtain acceptable results, it is necessary to use values of ***HSph. subdivs*** > 50.

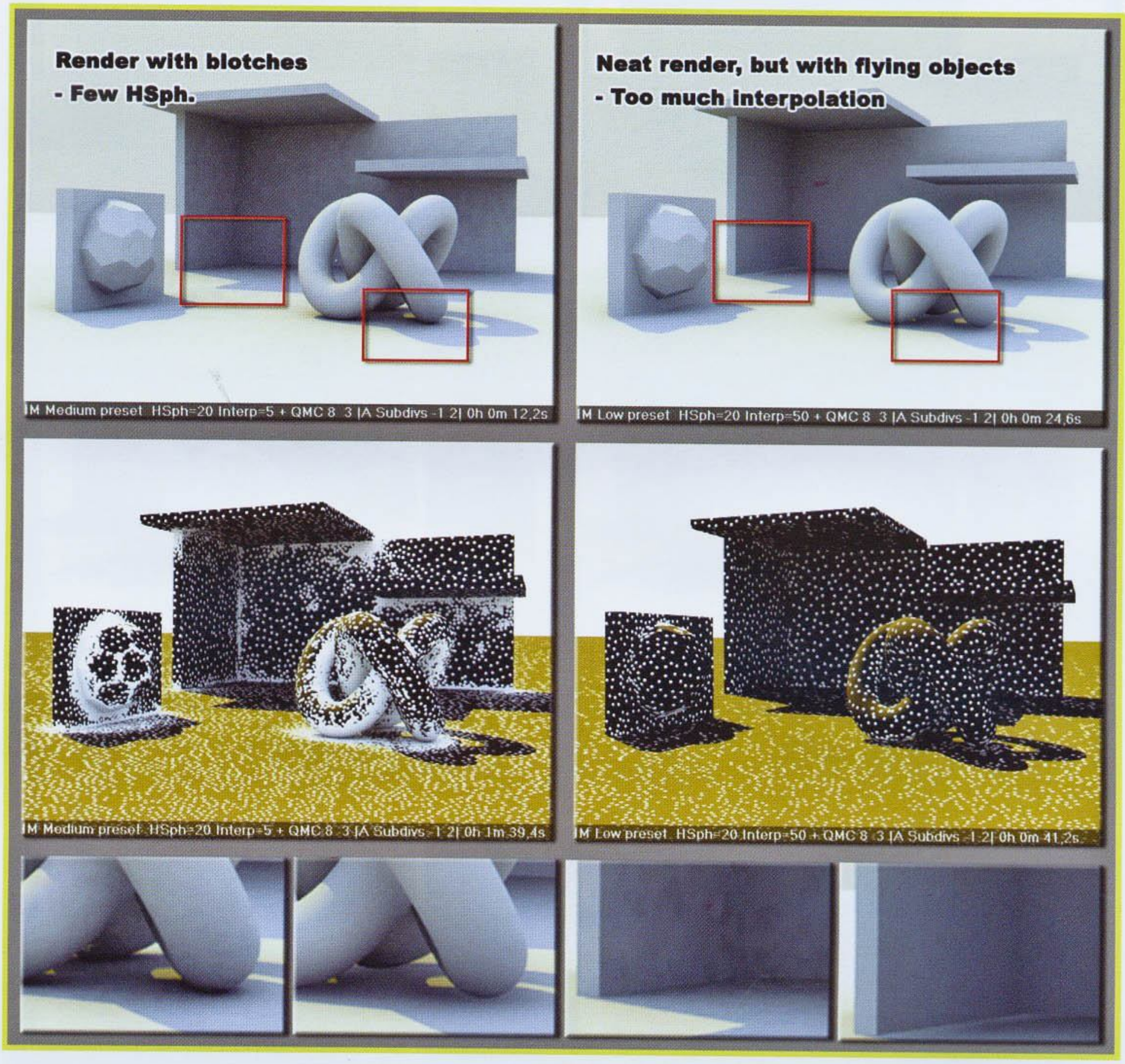
Interpolation samples (Interp. Samples) - this parameter controls the quality of interpolation between ***IM*** samples. With high levels of ***Interp. samples*** the image is cleaner, although there may be loss of detail. Low values produce more detailed images, but the render has good chances of being "blotchy". Rendering time will be less though.
The ***IM*** is a map made of many samples, all of which contain information regarding illumination, shaders etc… These samples do not, however, cover the whole surface of the model, but only specific zones, the most significative ones. What happens in the zones which have not been sampled?



■ **Figura 3.146**
The unknown quantity is the exact value of the GI in the points which have not been calculated in the ***Irradiance Map***.

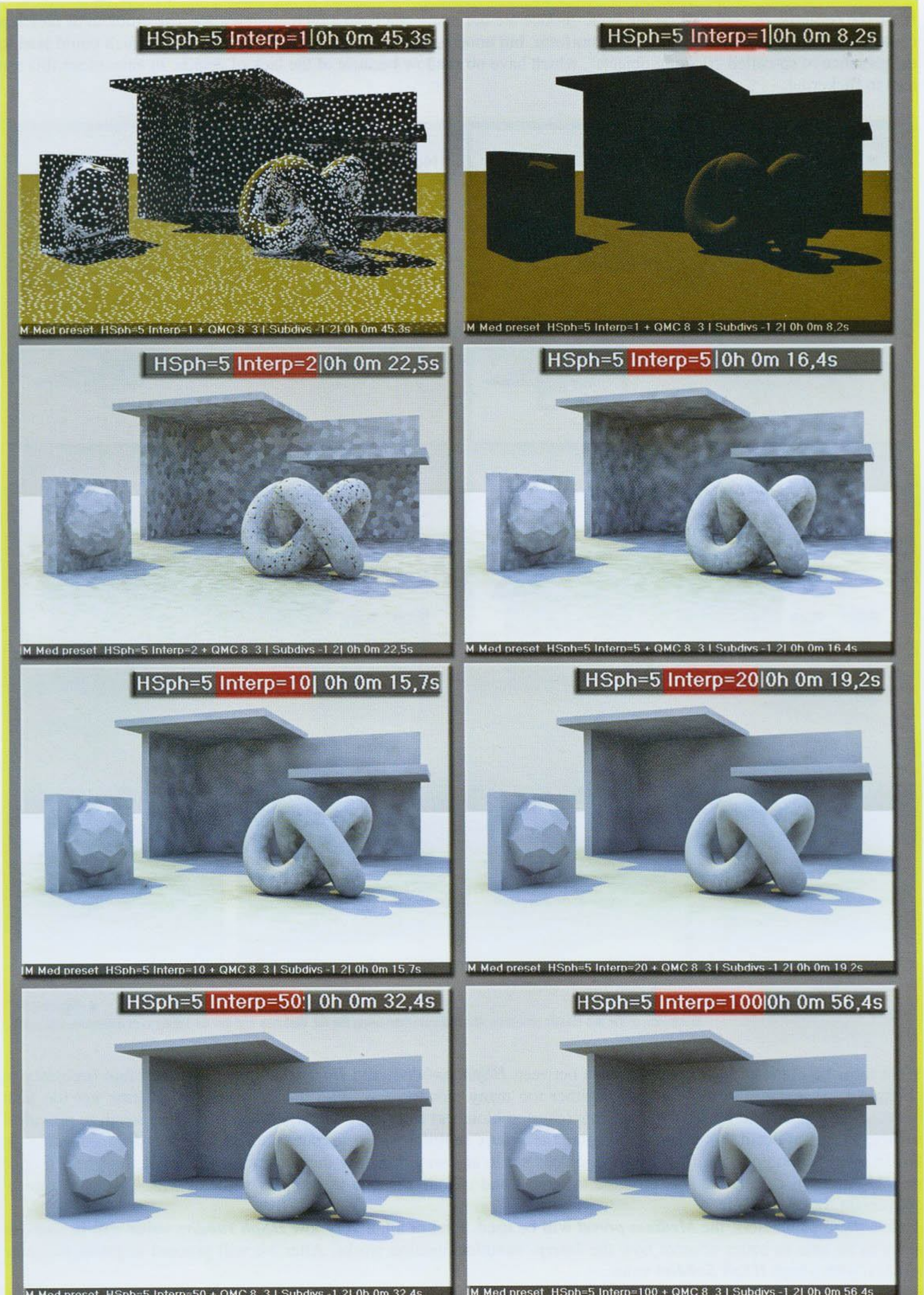
In the completely dark areas, the missing information is acquired by means of interpolation between nearby points. This means that the more samples are present on the surface, the better the approximation is. The more the samples, the closer their correspondence must be with the correct value. If this were not the case, the problem one would face, would be a "blotchy" render, because the samples, at the same level of ***HSph. subdivs***, would end up appearing to be closer to each other, and for a precise GI they must be as precise as possible. One can state that the more samples (white dots) are present, the higher the ***HSph. subdivs*** value should be and the lower the Sample Interpolation value can be.

If, on the contrary, the samples on the same surface are not many and have a high interpolation, the problem of blotches could be solved and the render would be uniform, but another problem will arise: a loss in detail. Which could lead to the presence of so-called "floating objects", which have no shadow because of the lack of details. In animations this can lead to flickering.

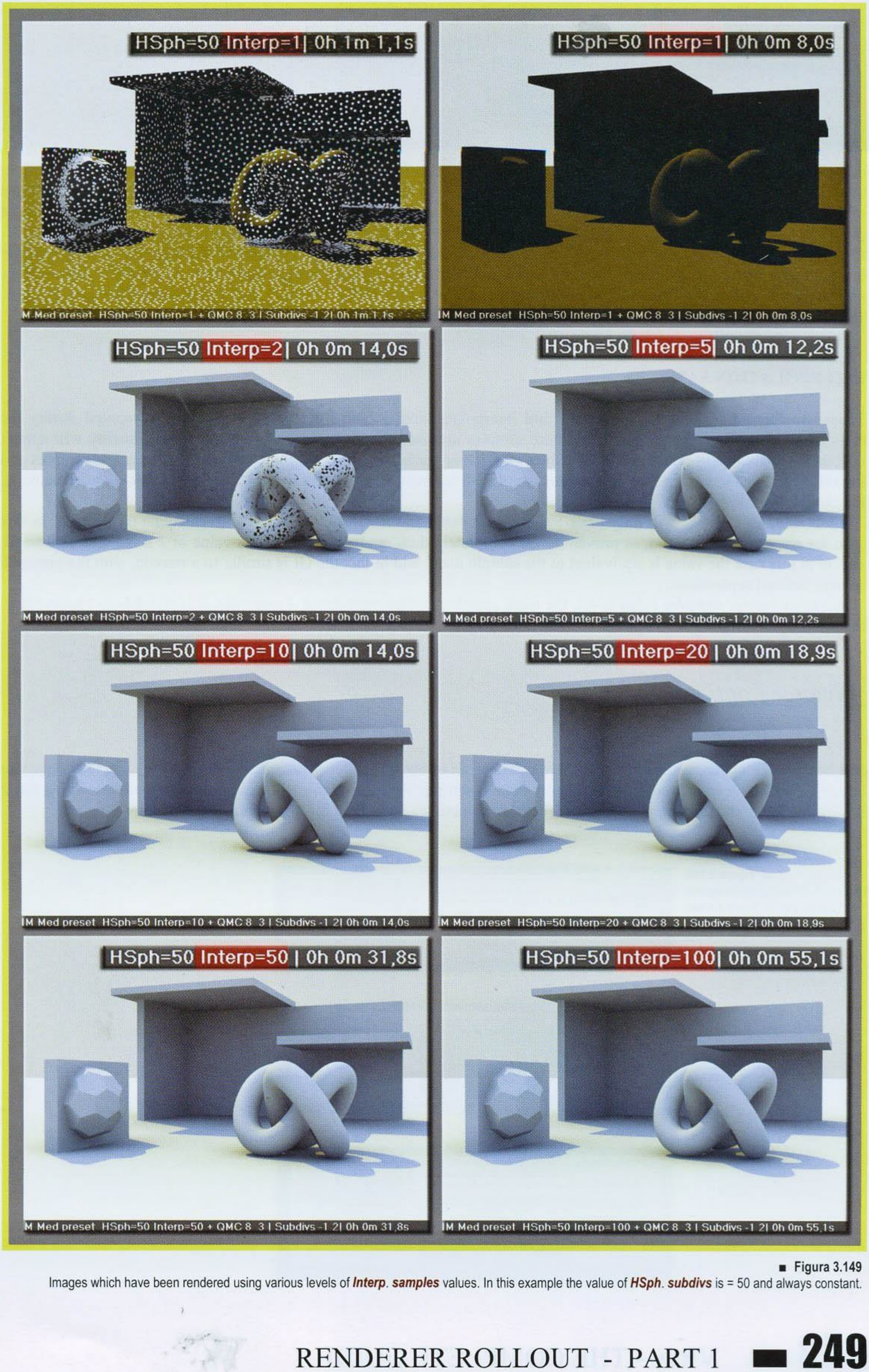


■ Figura 3.147
Illustration of the two classic problems which occur when using the ***IM***: Blotches and low GI definition or excessive uniformity.

What must be obtained is the right balance between ***HSph. subdivs***, and ***Interp. samples***. It is therefore necessary to obtain a sufficient quantity of samples; neither too many, nor too few, with the right level of quality: not too high (causing long calculation times), nor too low (causing blotches) and with an interpolation which is enough to provide a sufficient level of detail. Fortunately, changing the ***Interp. samples*** parameter does not require a new ***IM*** calculation. In fact, it takes action upon the final render and does not in any way influence the generation of the ***IM***. As a result, once the ***IM*** has been calculated, it is possible to load it and change the ***Interp. samples*** value at will. This is extremely convenient for saving time.
To test these parameters, the ***Medium preset*** will be used. First of all, a very low ***HSph subdivs*** value will be used, in order to be able to better observe how the ***Interp. samples*** function works. After we will proceed to fulfill the same renders, with a high ***HSph Subdivs*** value.



■ **Figura 3.148**
Images which have been rendered using various levels of ***Interp. samples*** values. In this example the value of ***HSph. subdivs*** is very low and constant.



■ **Figura 3.149**
Images which have been rendered using various levels of ***Interp. samples*** values. In this example the value of ***HSph. subdivs*** is = 50 and always constant.

The images in the previous examples show a simple scene rendered with different ***HSph. subdivs*** and ***Interp. samples*** values. Both have been submitted to the same variation of the ***Interp. samples*** parameter, but in the first image ***HSph. subdivs*** = 5, a very low setting, instead in the second image it is set to a more correct value, usually used for medium quality renders, that is ***HSph. subdivs*** = 50.

HEMISPHERE SUBDIVISION

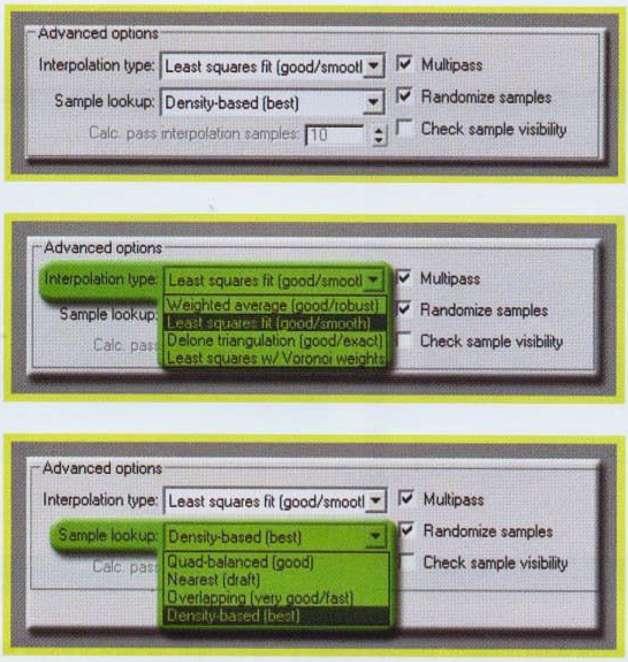
This parameter controls the quality of each and every sample. The position of samples is not connected to this value, as demonstrated in the previous examples. Only on the curved surface are there evident differences in the distribution. In the other areas the quantity and position of samples is more or less identical. This is due to the fact that a very low value of ***HSph. subdivs*** has been used, beyond any feasible purpose. In fact a value of 5 has been used only for demonstrative purposes. If we had applied a value whereby ***HSph. subdivs*** > 20, this would not have happened.
If one then decides to analyze the two different versions, it is immediately noticeable how the quantity of rays used can bring about a better GI quality. Even with very low interpolation values (***Interp. samples*** = 5) the GI is already very good. Default values for the ***HSph. subdivs*** setting are in fact set to 50. Usually, for trial renders, one uses a ***HSph. subdivs*** value equivalent to 20, whereas for final renders one can even go above 100 ***Subdivs***.

INTERPOLATION SAMPLES

Thanks to this parameter, one can cover and interpolate areas which have not been directly computed during the elaboration of the ***Irradiance Map***. The more samples are used for interpolation, the more the GI quality will appear soft, as the average will tend to level GI values on a larger surface, instead with low values *VRay* uses less samples and the GI is more detailed. This means that the samples are correct, in other words the higher the ***HSph. subdivs*** value is, the cleaner the GI is.
With a value of 1, *VRay* will only have 1 sample available and it will not be possible to calculate an average. In this case the operation is not valid, as proven by the completely dark render result. With a value of 2 it is possible instead, except in this case the value is equivalent to the sample itself, and in fact the GI is similar to a mosaic, with the presence of very defined separations.
The higher ***Interp. samples*** value is set, the more samples will be considered. From tests, it is possible to observe how the optimal value runs within a range of 10 and 20. Higher values do not give relevant improvements.

Advanced options

Just beneath the ***Basic parameters*** box, there is a new set of options called "***Advanced***". With these parameters one can have an even better control of the ***IM***. Usually default settings are more than sufficient to obtain excellent quality, but it is useful to know these parameters because they can sometimes be of help for solving certain problems.



■ Figura 3.150
The panel containing the advanced settings for the ***IM***. There are two pull-down menus and four options.

Interpolation type - in this pull-down menu one can select a series of parameters. These options are used only during the final rendering process and do not intervene during the calculation of the ***Irradiance Map***. They can therefore be modified at will, simply by loading the ***IM*** which has been saved beforehand (shortly this will be explained). These parameters allow one to decide which type of interpolation used by ***VRay*** to load the areas which are not calculated in the ***IM***, the completely dark areas.

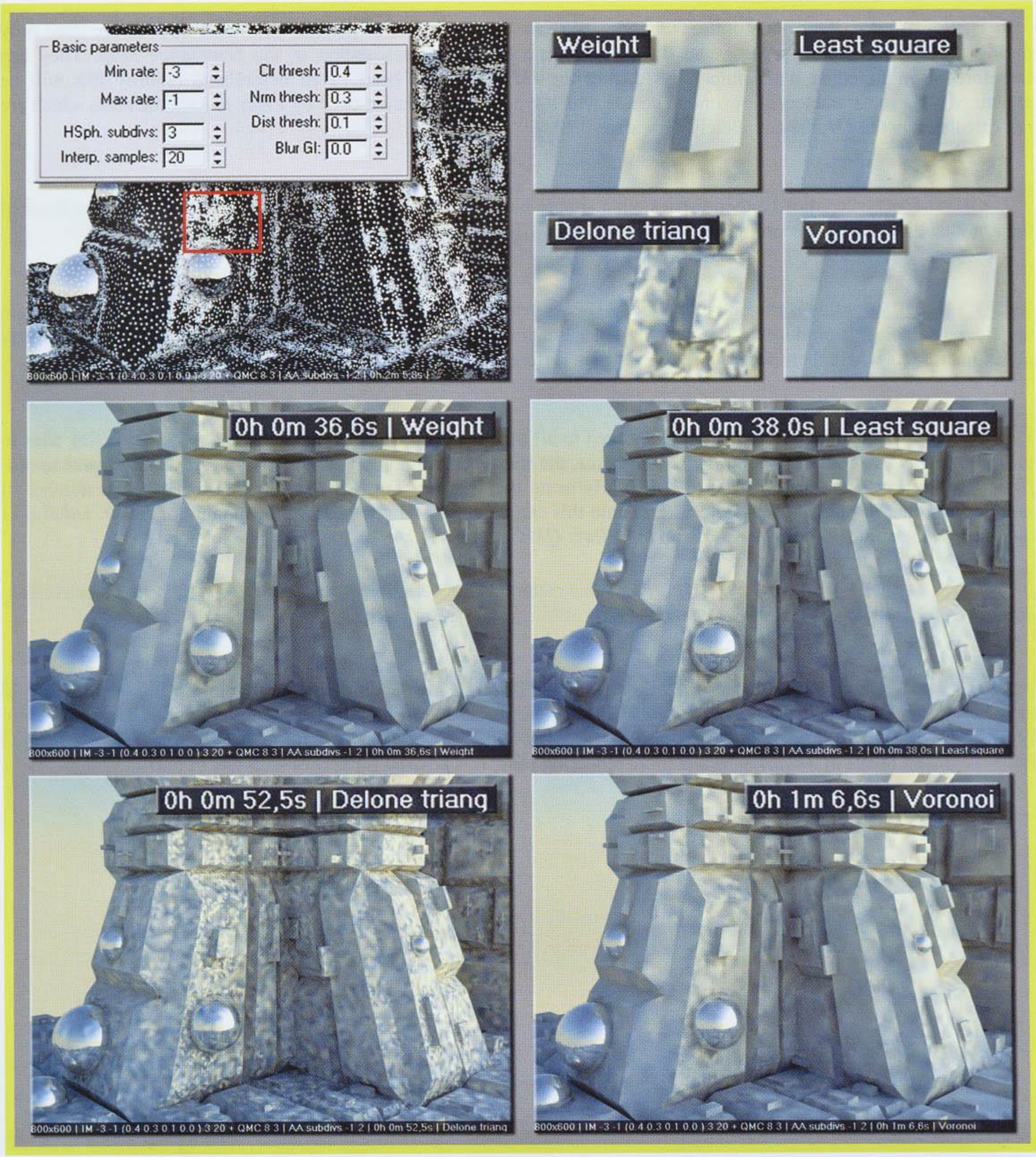
Weighted average - this method carries out a simple bleeding (mixing) between samples in the ***IM*** on the basis of their mutual distance and according to the surface normals. It is a fast method but tends to produce blotches.

Least square fit - this is the default method. This system computes the GI by searching for the best value between the closest samples. It produces softer and smoother images compared to ***Weighted average***, but I also slower. It may generate artifacts where there are samples with strong differences in contrast and density, such as rings and halos.

Delone triangulation - this method, compared to all the others, is a "non-blurry" method, whereby ***VRay*** tries to preserve details. Like all other "non-blurry" systems, the result can contain pointlike noise. Systems like ***Least square fit*** or ***Weighted average*** instead tend to hide it. By using this method there is in fact a need for a greater number of samples, in order to obtain a satisfactory result and this can be obtained by increasing the amount of ***HSph. subdivs*** or diminishing the value of ***Noise threshold*** in the ***VRay: QMC sampler***. Later on, this will be explained.

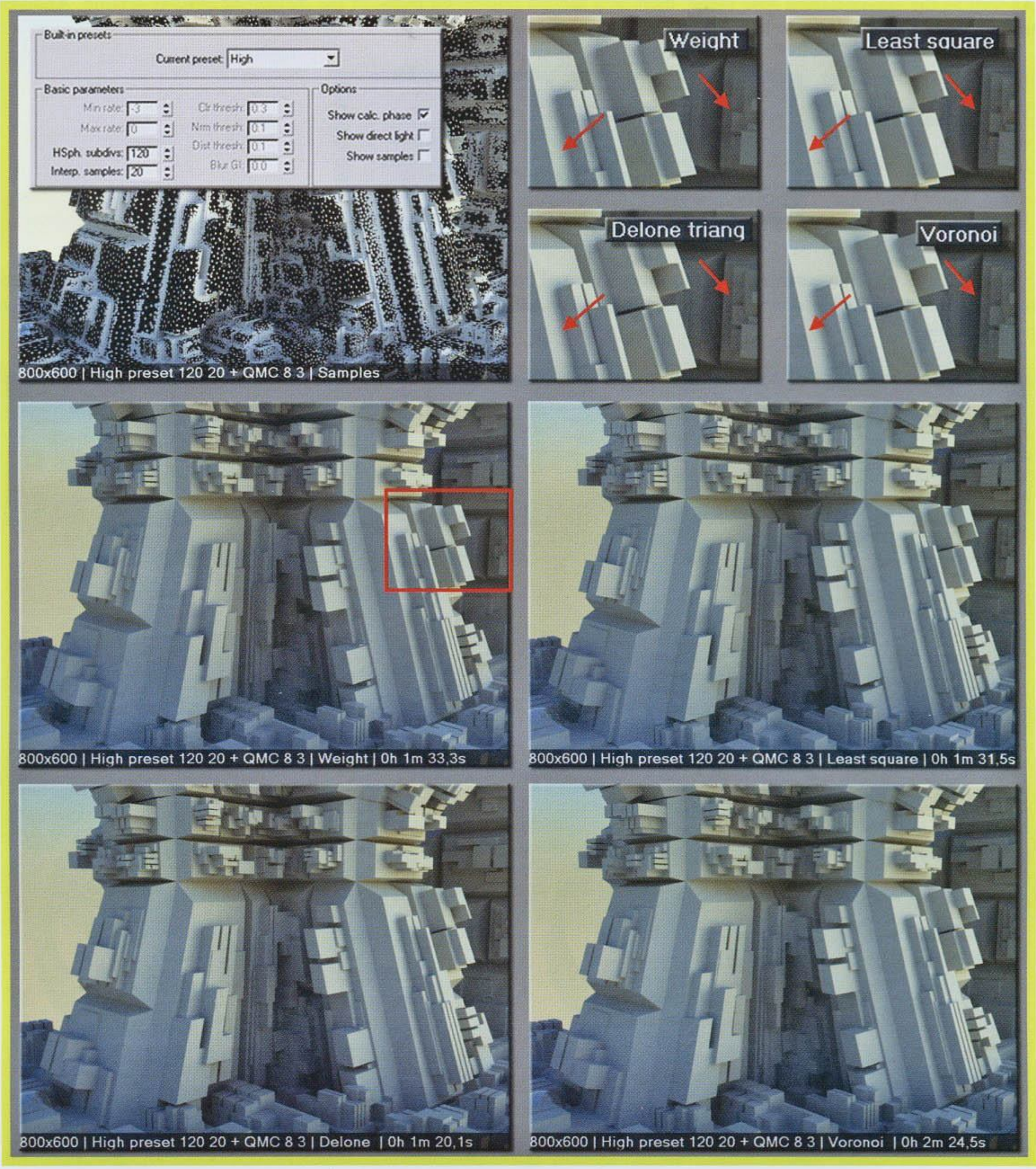
Least square fit with Voronoi weights - this method is the variation of the ***Least Square fit***. Its objective is to eliminate problems. It is quite a slow method and its efficiency is questionable.

All these methods have characteristics that make them interesting in certain circumstances, but probably the most useful are ***Least square fit*** and ***Delone triangulation***. The former, because of its intrinsic nature, tends to hide noise, losing detail. It is perfect for scenes with vast uniform areas. ***Delone triangulation*** is a so-called "exact" method, which requires greater levels of ***Subdivision*** and more rendering time. It is advised for scenes with lots of details.



■ **Figura3.151**
The four interpolation methods available in VRay.

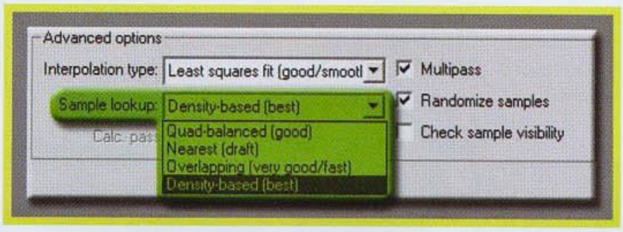
In the previous test a scene rendered at a resolution of 800 x 600 pixels was used. Very low ***IM*** values were used. This way it is easier to observe the differences between the various interpolation methods. For each of the four tests the same parameters were used.
The most evident difference is seen in the test carried out with the ***Delone triangulation*** method. With all the other methods ***VRay*** tends to give a more homogeneous GI, instead with the ***Delone*** method this does not happen. It is so precise that in some spots one can notice the grid of samples generated by the ***IM***. The remaining methods, instead, mingle the samples, making the GI cleaner and more uniform, but with a tendency to lose details.



■ **Figura 3.152**
The four interpolation methods available in VRay, applied with very high ***Irradiance Map*** values. The model is much more detailed.
- - - DVD - - -

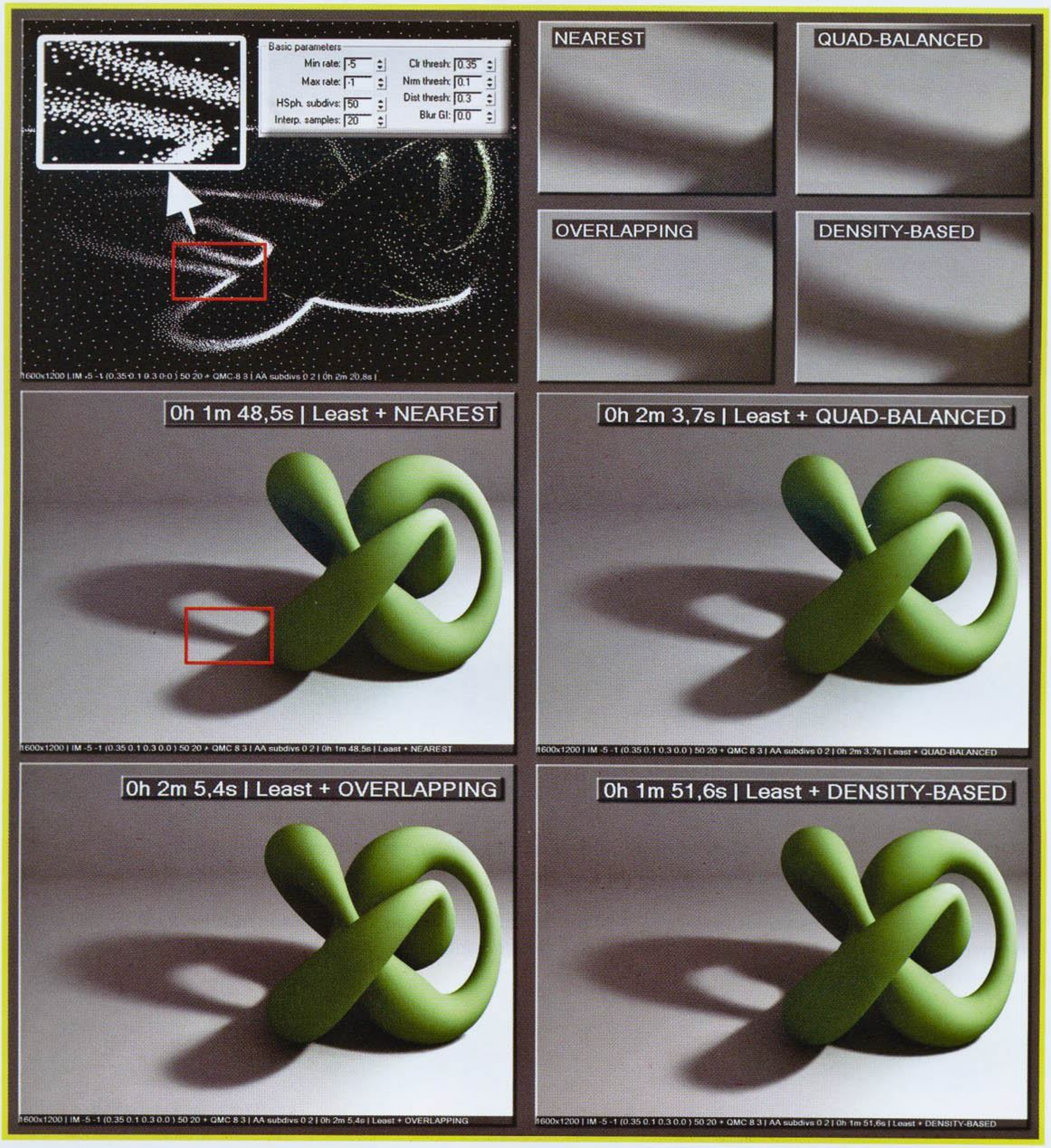
This time very high settings values have been used for the ***IM***. As one can see, the difference between the methods is less obvious. There are however some zones of the render which are better, both in terms of GI and of detail preservation, when using ***Least square fit***.

As well as the ***Interpolation type*** menu, there is a second pull-down menu in the ***Advanced options*** section. Another tool for managing the ***IM*** is to be found in the ***Sample lookup*** options.



■ **Figura 3.153**
The pull-down menu for the ***Sample lookup*** parameter.

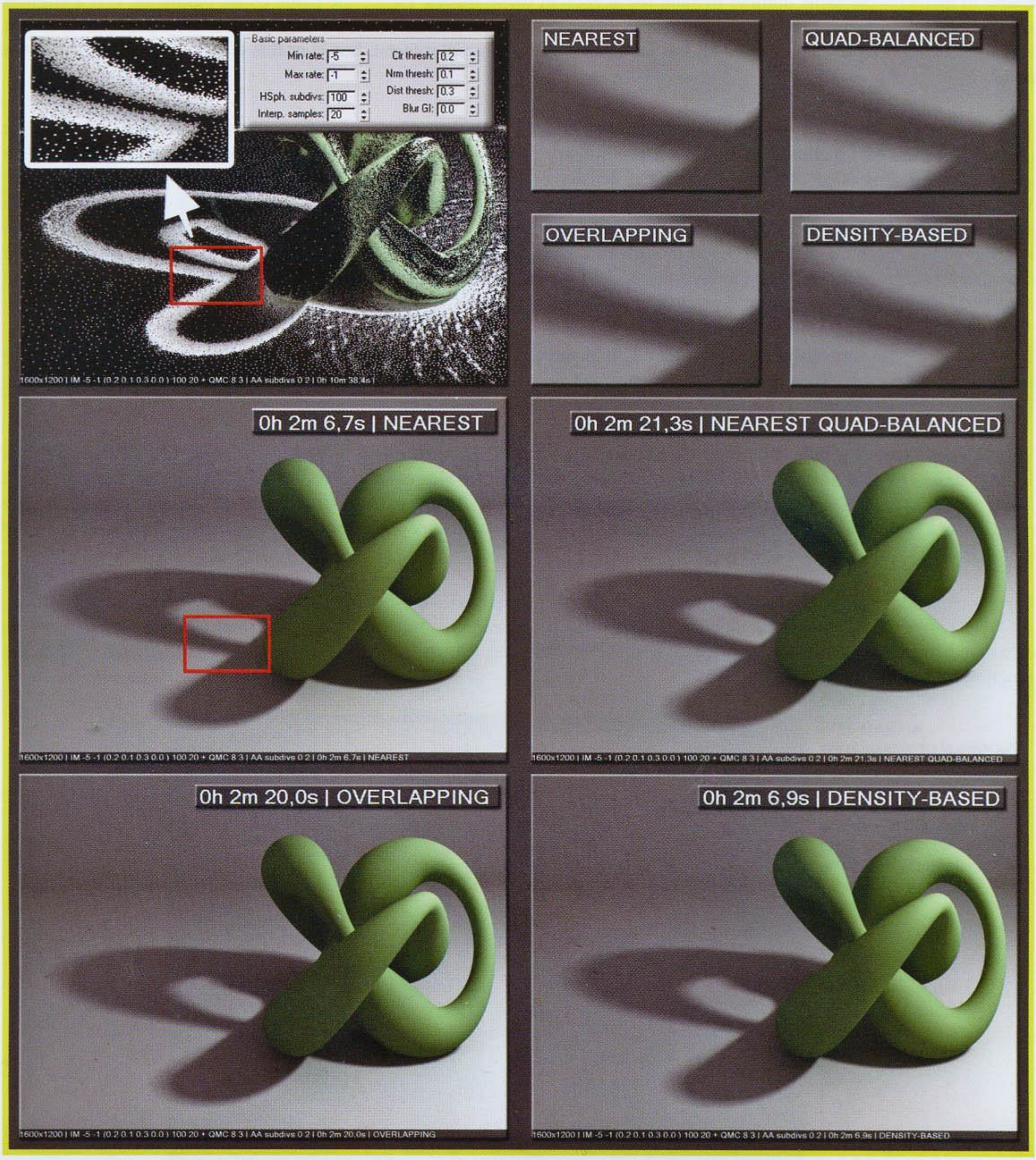
Sample lookup - the parameters in this menu are used only during the final rendering. The methods in the pull-down menu are used to decide which of the many possible samples located around a specific point are to be used for interpolation.



■ **Figura 3.154**
The four ***Sample lookup*** methods in VRay applied with medium values in the ***Irradiance Map*** settings.

This last test involved a scene rendered at a resolution of 1600 x 1200 pixels. A lot of attention has been given to zones where light passes to dark or vice versa. It can be seen how the ***Overlapping*** method, in this case, eliminates almost any defect. But it is also the one which blurs the GI the most. In this case the ***IM*** values, specifically ***HSph. subdivs***, are set to medium level.

In the next example high ***IM*** values are used, and similarly to interpolation methods, also in this case the difference between the various methods of ***Sample lookup*** are greatly reduced. There are very slight artifacts which can be seen only from nearby. In any case, the best results are obtained with the ***Overlapping*** method.



■ Figura 3.155
The ***Sample lookup*** methods available in VRay, applied with high ***Irradiance Map*** settings values. The difference between the four methods is very little.
- - - DVD - - -

Nearest - this method chooses which of the samples to use are closest to the area to interpolate, via the ***Interp. samples*** parameter, seen earlier. It is the fastest method, and in past versions of *VRay*, it was the only one available for a long time. The problem with this system arises when there is a big difference in density between samples. In some zones there is a big concentration of samples, as opposed to other areas which are less dense. When a blurry interpolation method is used, such as ***Least square fit*** or ***Weighted average***, a phenomenon called ***density bias*** can occur. In this case, problems could arise, due to an incorrect interpolation precisely in one of those high density areas, like for example when using ***VRayShadow***.

Nearest quad-balanced - this method is an extension of the ***Nearest*** method and was studied to avoid the abovementioned problems. What it does, is divide the space around the sample to be interpolated, into four sections, trying to maintain the same number of samples in each of these spaces. Hence the name ***Quad-balanced***. It is a slightly slower method compared to the preceding one, but usually produces much better results. The only problem is that it sometimes uses samples which are a bit too far away and not of great interest, however this does not influence the result much.

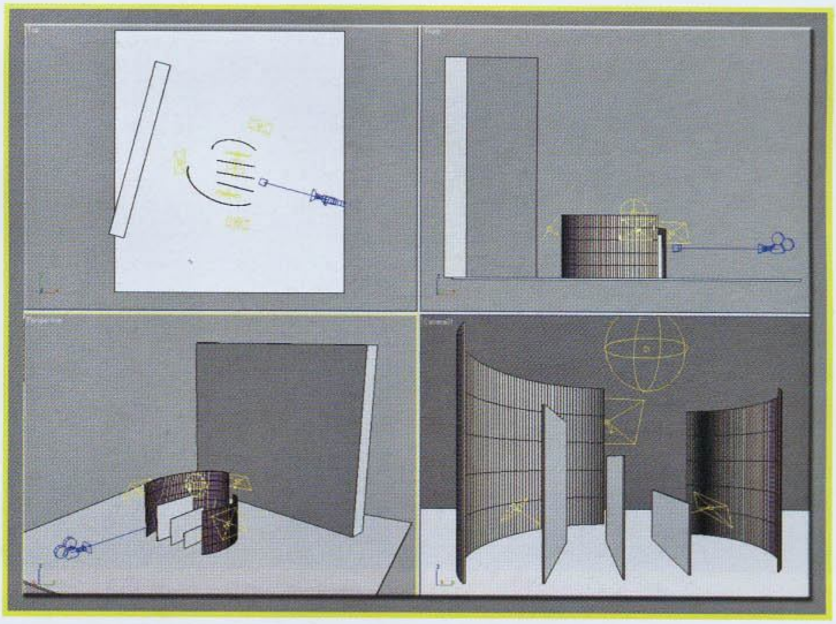
Precalculated overlapping - this method was created with the precise aim of avoiding the problems which take place in the first two methods. It requires a preliminary pass, where *VRay* calculates the radius of influence for each sample. This radius is greater, the smaller the density of samples is. The greater the density, the smaller the radius. When one has to calculate interpolation, *VRay* chooses the samples within the influence of this radius. One of the advantages of this method, is that when used together with a blurry interpolation method, a continuous and uniform GI is generated (smooth GI solution). Even though an additional step is necessary with this method , it is often faster than the other two. It is ideal for high quality images. Unfortunately it sometimes uses far away and unimportant samples, as a result altering the final quality and producing a blurry GI.

Density-based - this is the default method. It is the amalgamation of the ***Nearest*** and ***Precalculated overlapping*** methods and it is very efficient in the reduction of artifacts. It requires an initial pass, as for ***Precalculated overlapping***, but recovers quality thanks to the fact it uses the ***Nearest*** method.

The ***Nearest*** method is advisable only for previewing, whereas ***Nearest quad-balanced*** generates good quality images in most cases. ***Precalculated overlapping*** is also very fast and provides excellent results, although it tends to generate a blurry GI. Lastly, the ***Density-based*** method is the one that is apt in most cases. It is to be remembered that these ***lookup*** methods are important especially when using blurry interpolation methods. When using ***Delone triangulation***, these methods hardly influence the final result.

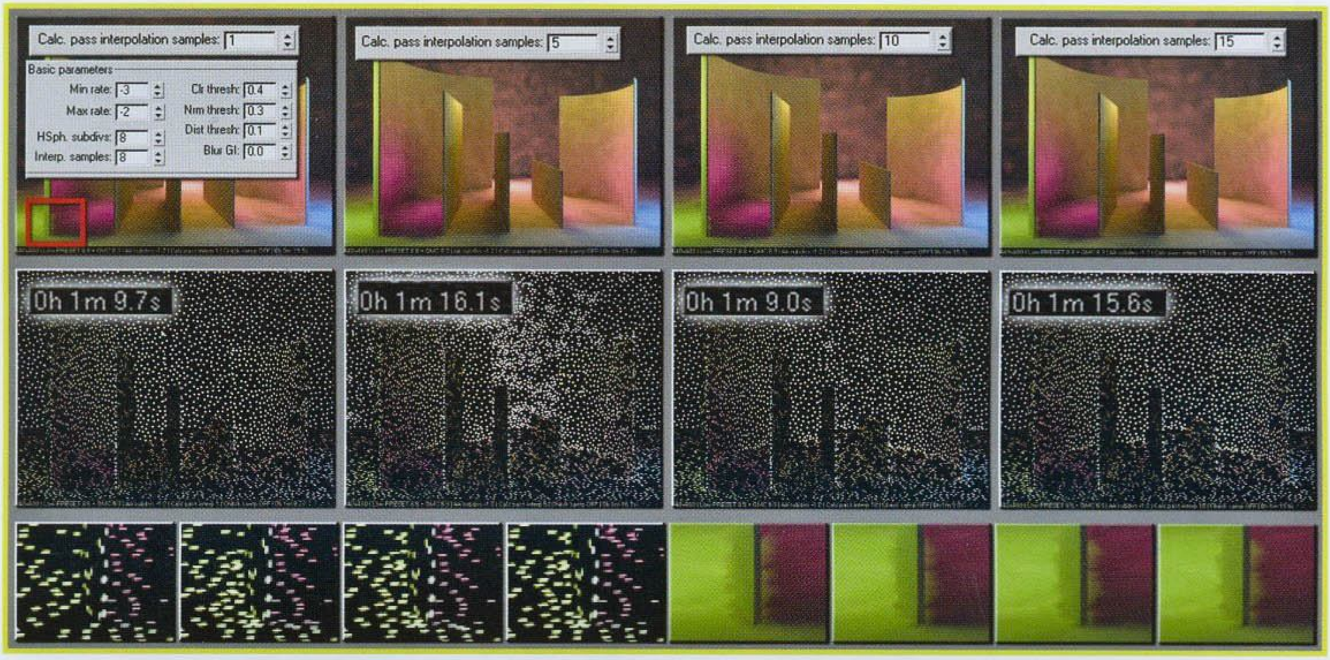
For the next four parameters, a very simple scene is used. It is made up of seven polygonal geometries, illuminated by six light sources, each one with the ***Store with irradiance map*** parameter active. This way ***Area Shadows*** will be calculated by the ***IM*** and not via ***QMC***. It will be seen later on, in the chapter dedicated to ***VRayLight***, what the function of this option is.

Figura 3.156
Four views of the viewport in 3ds Max. This scene is useful for explaining the following parameters: ***Multipass***, ***Randomize***, ***Check sample visibility*** and ***Calc. Pass interpolation sample***.

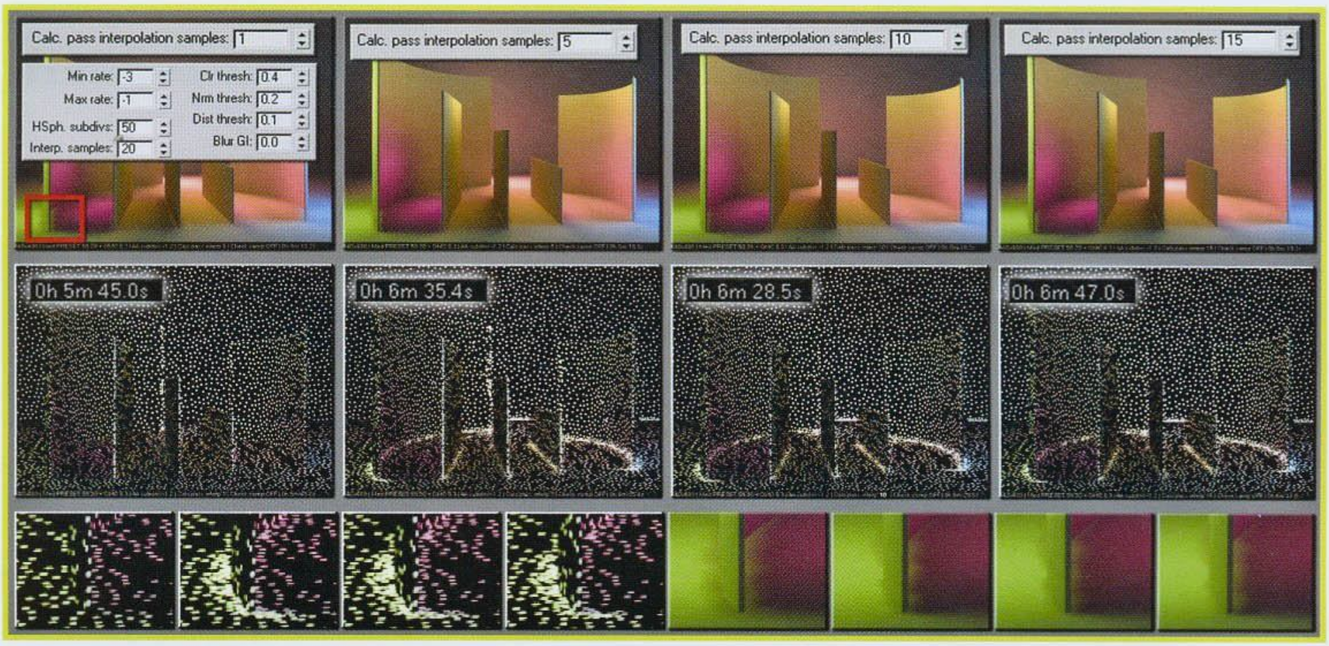


Calc. Pass interpolation samples - this is a value used for ***IM*** calculation, so changing it means recalculating the ***IM***. It represents the number of samples that the ***IM*** has already calculated and which will help the ***IM*** algorithm for the following calculations. Ideal values are between 10 and 25. Less than this means a faster ***IM*** calculation but may not be sufficient to obtain good quality images. High values make the ***IM*** calculation slow, by increasing sampling. The default value of 15 usually assures correct results.

In the case we are examining, the variation of the ***Calc. Pass interpolation samples*** parameter brings very few changes to the ***IM*** and the default values are more than enough for a correct GI calculation. In the next two examples, carried out firstly with ***Low*** and then ***High IM*** values, the ***Calc. Pass interpolation samples*** parameter has been set to 1 sample, then 5, 10 and lastly 15 samples. Only very few differences are noticeable both in terms of quality and rendering time.



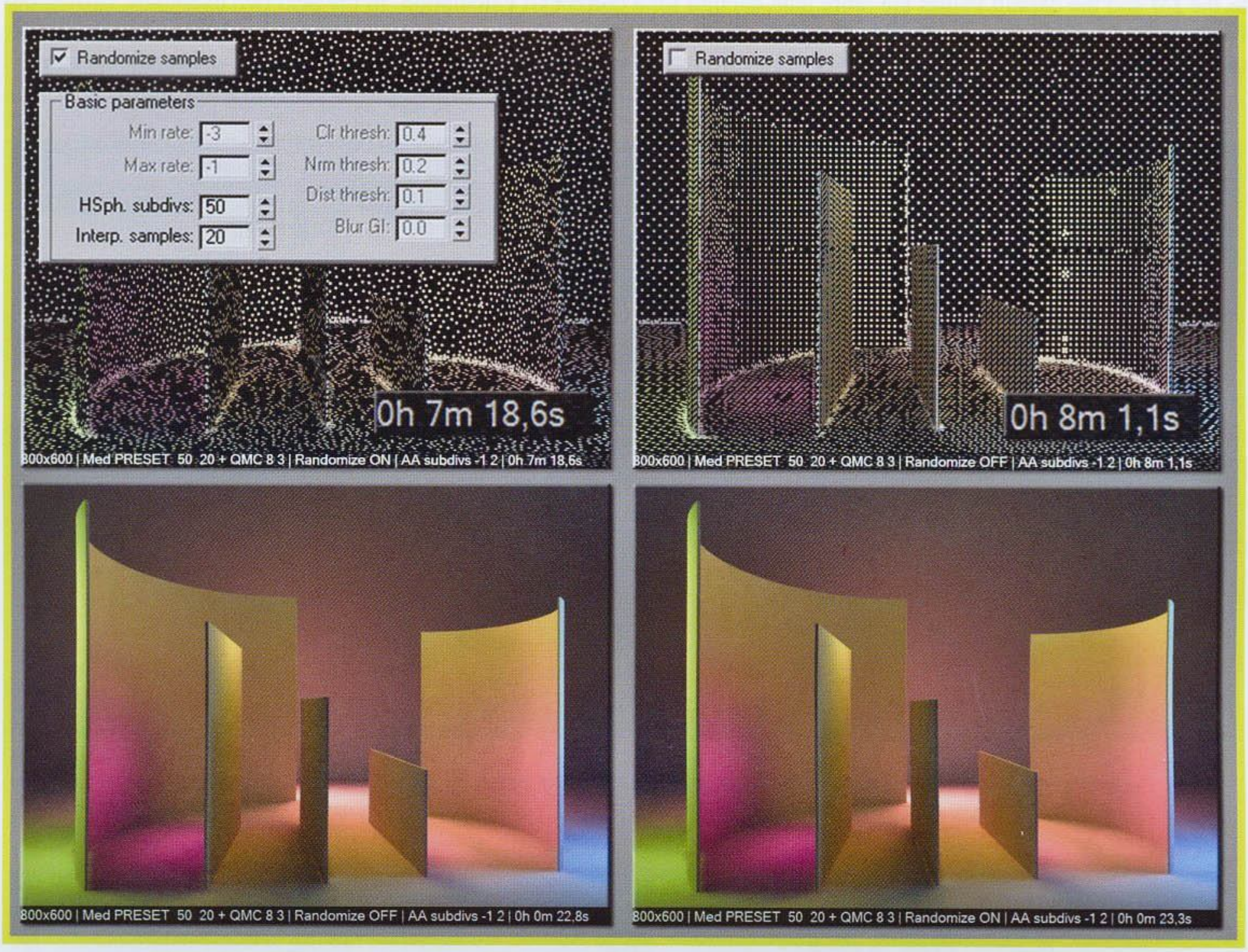
■ **Figura 2.157**
A scene rendered with values 1,5,10 and 15. the ***IM*** was calculated with low parameters, in order to emphasize the changes applied to the ***Calc. Pass interpolation samples parameter.***



■ **Figura3.158**
With high ***IM*** values the difference is even less.
--- DVD ---

Multipass - this parameter is used during the ***IM*** calculation. When activated, ***VRay*** uses the ***Prepasses*** set with ***Min rate*** and ***Max rate*** for ***IM*** generation. When disabled, the ***IM*** is calculated in a single passage and the calculation is slightly slower.

Randomize samples - this is used during ***IM*** calculation. When activated, samples are randomly generated. When deactivated, the distribution of samples follows a "grid" scheme. In general, it is advisable to leave it on, in order to avoid artifacts due to excessive regularity of samples. In the example high ***IM*** values are used. Using a normal sampling system, the result does not change much, compared to that obtained with the ***Randomize samples*** parameter activated. The only disadvantage is a small increase in rendering time.



■ **Figura 3.159**
One can see the difference in the distribution of samples: on the left it is quite irregular. On the right, instead, the grid is easily visible.

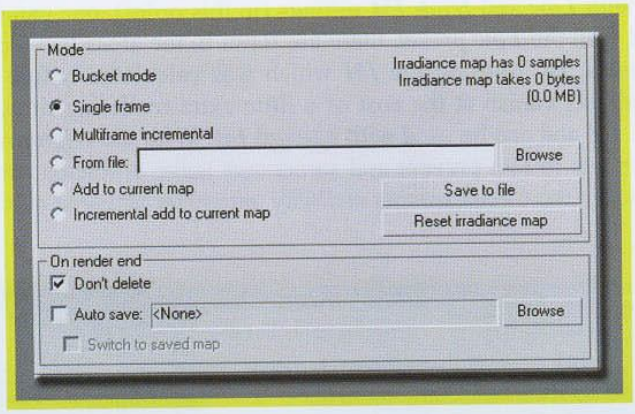
Check sample visibility - this is a parameter used during rendering. When it is active, only samples which are directly mutually visible to each other are used by *VRay* for interpolation. It is a very useful parameter to avoid problems caused by the interpolation of samples which are near each other but divided by a small surface such as a thin structure. Activating this parameter, however, takes up extra rendering time because *VRay* must use additional rays to detect wether samples are directly visible or not.
In the next few examples, rendered with ***Low*** and ***High IM*** settings (in this order), one can see the effect of activating this parameter. In the scene rendered with the ***Low preset***, there are flaws in the areas between the walls and pavement, white spots at the bottom of the panels. By re-using the ***IM*** which was calculated before and activating the ***Check sample visibility*** parameter, these artifacts vanish at the cost of a little extra rendering time. One must remember that this parameter does not influence the ***IM*** and can be used with a saved ***IM***. With the ***High preset***, these artifacts tend to disappear. A high number of samples can in fact prevent and sometimes solve problems caused by samples being too close. In any case, it is preferable to leave the ***Check sample visibility*** parameter active at all times.

■ Figura 3.160
With low ***IM*** values, at the base of the panels, light passes from one side to the other. The solution for this problem is to activate ***Check sample visibility***.

Mode

It has been mentioned more than once about the possibility of saving the ***IM***, in order to be able to use it again later on. Thanks to the next series of parameters, one can manage, save, unite and load the ***IM***.

■ **Figura 3.161**
Section of the ***VRay: Irradiance map*** rollout containing options for managing the ***Irradiance Map***.



The screen is divided into two zones: "***Mode***" and "***On render end***".
The first of these contains parameters which allow the creation of an ***IM***. It can be generated by regions (***Bucket mode***), generated for each single frame using one region (***Single frame***), loaded from a file saved before (***From file***) or one can accumulate the GI data in an animation. For animations three different calculation modes are available: ***Multiframe incremental***, ***Add to current map*** and ***Incremental add to current map***. Soon we will discuss the differences between these methods. Also, in the top-right corner of the screen, there is some information regarding the ***IM***, such as the total number of samples used, and the size in Megabytes.
The second panel contains parameters for managing the ***IM*** saving options. One can choose not to erase it from the memory once it is generated (***Don't delete***), or define the auto-save path (***Auto save***). One can also assign ***IM*** auto-save at the end of a render and set it to load automatically (***Switch to saved map***).

Bucket mode - by using this calculation method, ***VRay*** divides the image into small regions called ***buckets***, which the user can see. For each region, which can be changed in size via some parameters in the ***VRay: System*** rollout, ***VRay*** calculates an ***IM***, as if it was an isolated frame. This is a useful system for distributed rendering, where each ***bucket*** is rendered by a single computer in a network. As the image is made up of many images, ***VRay*** has to carry out an additional calculation in order to unite the ***buckets***. As each of these ***buckets*** is independent from the others, it is necessary to remove all artifacts from the borders of each one. This requires a long rendering time. Even this way, small problems can occur along the borders anyway, this can be solved only with high ***IM*** values (***preset*** on ***High***, high ***HSph. subdivs*** values or lower ***Noise threshold***). With the last versions of ***VRay***, one can render in a network also with all the other available modes. This method is therefore mostly unused.

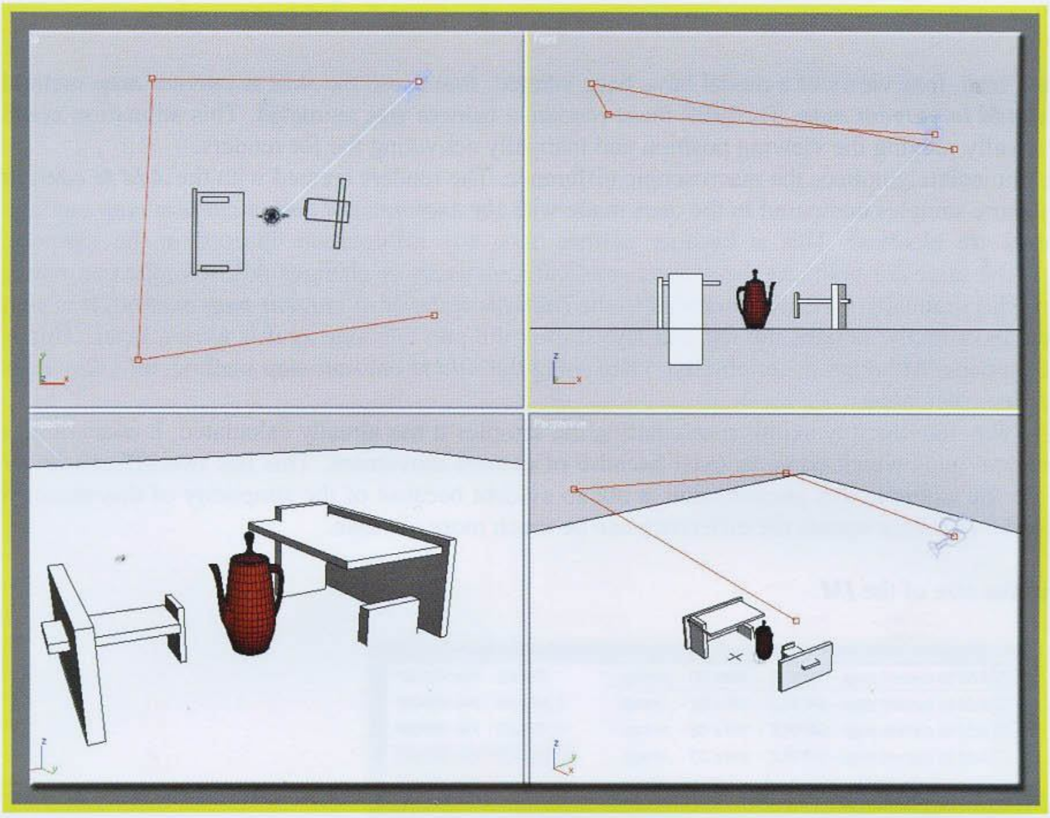
Single frame - this is the default method. The ***Irradiance Map*** is calculated for the whole image. In distributed rendering, each computer renders a piece of the image and contributes towards the final generation of the ***.vrmap*** file. Useful for single static renders or dynamic animations, both indoor and outdoor types.

Multiframe incremental - useful method when one wants to render a sequence of frames, which need not necessarily be consecutive, with a moving camera. This type of animation, where only the point of view varies, is usually called ***Fly-through*** animation. Other alterations in the scene, such as moving objects, changes in lighting or color/shaders are not contemplated in this type of method, making it useless.
With this system, once animation rendering is launched, ***VRay*** calculates an ***Irradiance Map*** for the first frame and eliminates anything which was held in the memory until then. ***VRay*** does not consider the ***IM*** of the previous render and neither the loaded one. For the following frames, ***VRay*** keeps adding new samples in the areas which have not yet been rendered before. In areas which have already been calculated, ***VRay*** re-uses the samples held in the memory, which saves a lot of time. This system for ***IM*** generation when there are multiple static views or ***Fly-through*** animations.

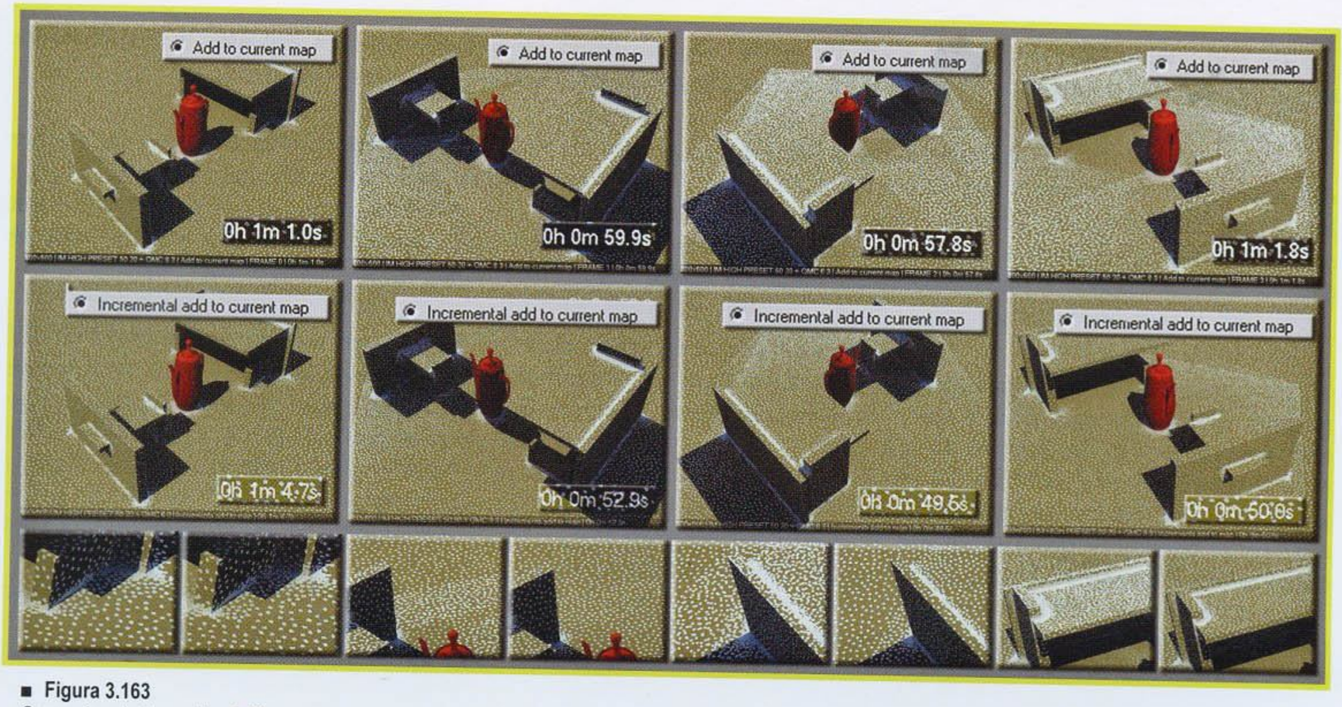
From file - with this mode, *VRay* loads the previously calculated ***IM*** into the memory and uses it for the whole animation, for each frame. This method can be used in ***Fly-through*** animations and network rendering.

Add to current map - with this method *VRay* calculates the ***IM*** of the current frame independently of any other ***IM***, adding it to the previous frame saved. For all the following frames, a new ***IM*** is to be calculated, no matter what has been calculated before. The use of this method could be defined as a sum of ***IM*** calculated in ***Single mode***. It is not a very productive method, as there is a certain consumption of energy and resources in the case of ***Fly-through*** animations. If necessary it is preferable to use the ***Incremental add to current map*** mode.

Incremental add to current map - this method is an improvement of the previous one. *VRay* calculates the ***IM*** of the current frame with the information obtained form the rendering of the previous frame which is held in the memory. During an animation, if new portions of the scene, which have not been calculated yet, are to appear, *VRay* adds new samples to the RAM in the ***IM***. Vice versa, if scenes which have already been calculated appear, *VRay* does not compute them a second time. At first glimpse, this could seem identical to the ***Multiframe incremental***, but there is a substantial difference. For the first frame in the animation, ***Incremental add to current map*** can use the ***IM*** in the memory or previously saved at the end of another session. This feature is not available in the ***Multiframe incremental*** mode, which eliminates anything regarding ***IM*** which is left in the memory.
This system is useful for generating ***IM*** for multiple views, both in static scenes and ***Fly-through*** animations, and for recovering the calculation of an ***IM*** of frame or animation which is still incomplete or which has been interrupted.



■ **Figura 3.162**
Screenshot from 3ds Max. The scene is made of mere primitives. The only moving object is the camera, which has four position keys which can be seen through the four squares at the angles of the red line. The aim is to render 4 static views with a camera placed at the four angles of the scene.

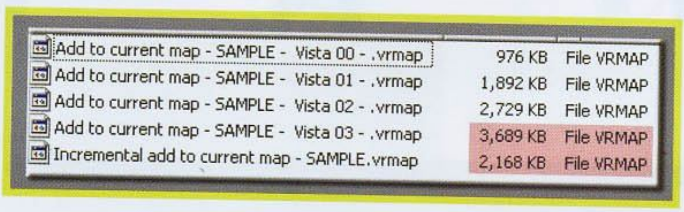


■ **Figura 3.163**
Comparison between the ***Add to current map*** method and the ***Incremental add to current map*** method.

In the example we have just seen, four views of a model have been created, first using the ***Add to current map*** method and then with ***Incremental add to current map***. To fulfill these renders a camera was animated. This animation could have been done also by manually moving the viewing position and manually activating the for renders.
Looking at the image, one immediately notices the macroscopic difference. The renders created with the ***Add to current map*** method contain many more samples compared to the ones made with the ***Incremental add to current map*** method. In the first view the results are identical. This is because neither have any information charged in the memory. Therefore the calculation is the same for both. As the camera gradually advances or changes position, one can notice that in the first case the samples gradually increase compared to the ***Incremental add to current map*** method. It is as if *VRay* calculates every view indistinctly, obtains the ***IM*** and lays it over the previous one as if it were a layer. This is actually more or less what happens. Although not explicitly, when using the ***Add to current map*** method, all *VRay* does is unite all the maps as they are calculated.
Instead in the second case, *VRay* intelligently avoids recalculating the samples it has already calculated. It concentrates on filling the empty spaces and gaps which come to exist because of camera movement. This has two effects: lower rendering time is the first. In the example this phenomenon is not so evident because of the simplicity of this scene. It diminishes only by 10 seconds. In indoor scenes the difference can be much more obvious.

The second effect concerns the size of the ***IM***.

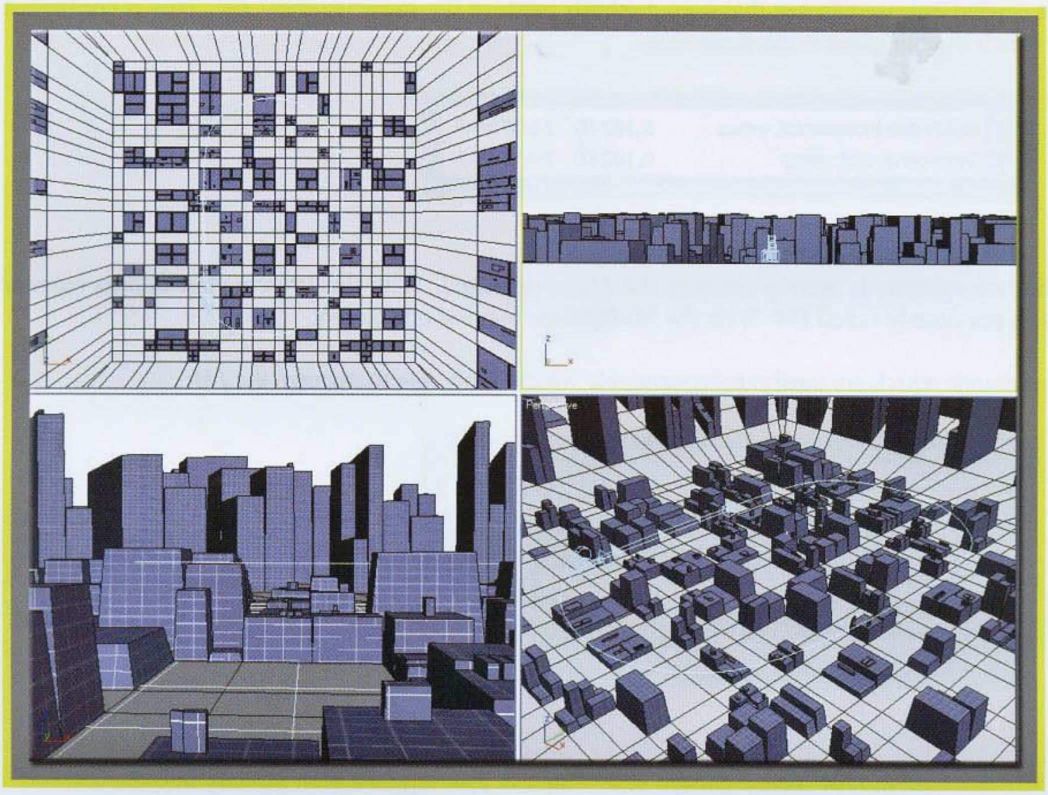
■ **Figura 3.164**
Progressive sizes of the ***IM*** for the ***Add to current map*** method and ***IM*** size for the ***Incremental add to current map*** method.



Add to current map - SAMPLE - Vista 00 - .vrmap	976 KB	File VRMAP
Add to current map - SAMPLE - Vista 01 - .vrmap	1,892 KB	File VRMAP
Add to current map - SAMPLE - Vista 02 - .vrmap	2,729 KB	File VRMAP
Add to current map - SAMPLE - Vista 03 - .vrmap	3,689 KB	File VRMAP
Incremental add to current map - SAMPLE.vrmap	2,168 KB	File VRMAP

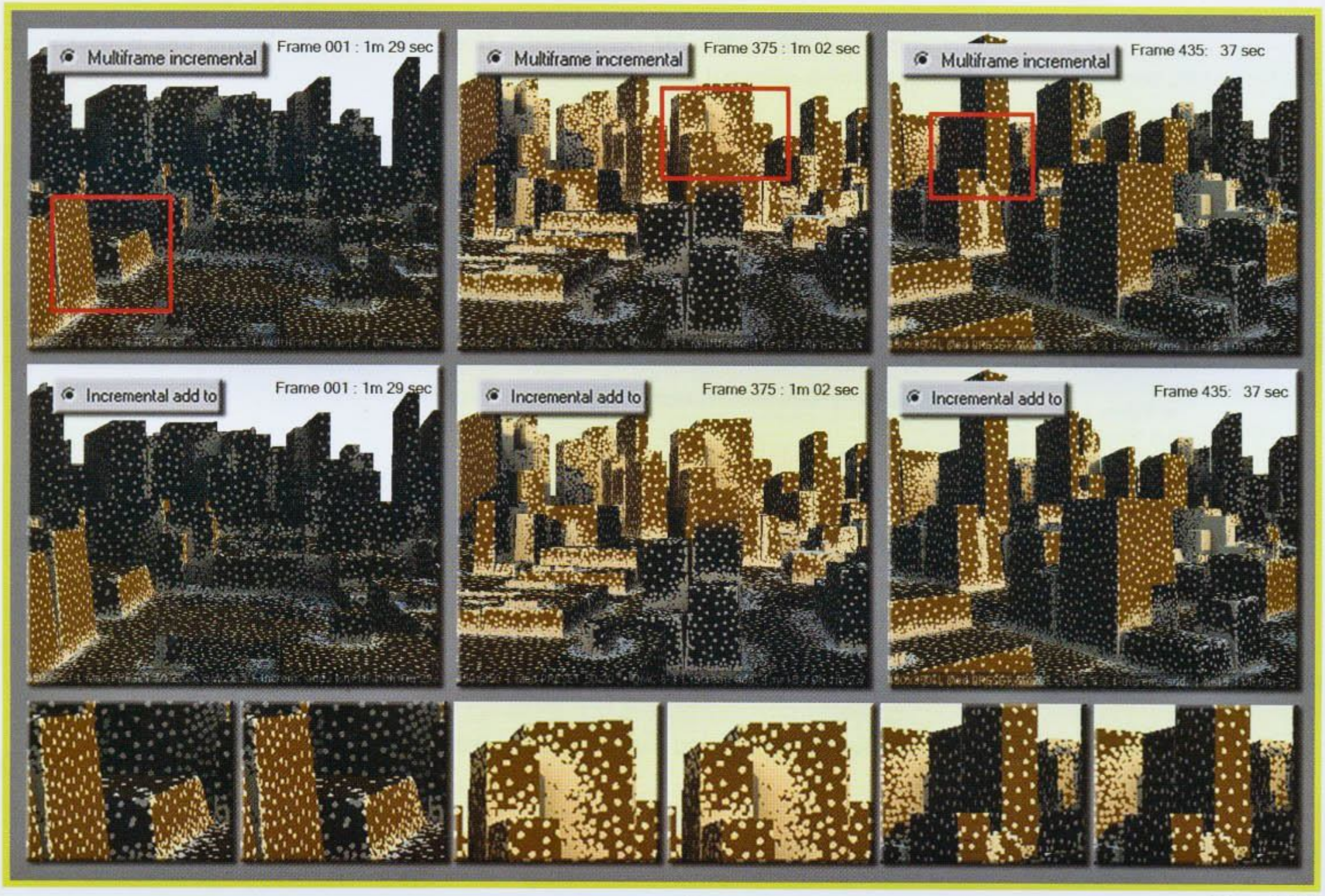
As one can see, the final ***IM*** generated by the ***Add to current map*** method is 40% larger than the one created with the ***Incremental add to current map*** method. This is certainly a significative difference. Obviously the ***IM***, being larger will be more precise, but this precision is often redundant and unnecessary. For large-size scenes, this optimization is definitely quite an advantage.

In the next scene we show how the ***Multiframe incremental*** method is identical to the ***Incremental add to current map*** method. Both produce the same ***IM*** with the same procedure and the result is of the same size. The scene to be used portrays a group of buildings. The animation is a simple loop of 500 frames, within the metropolitan area.



■ **Figura 3.165**
Screenshot from 3ds Max. The model is made of a single plane with a few extrusions produced with the ***Greeble*** plugin.

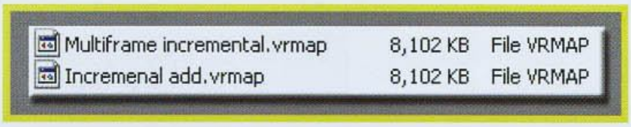
The ***IM*** was calculated both with the ***Multiframe incremental*** method and the ***Incremental add to current map*** method. These are the results:



■ **Figura 3.166**
Rendering of some frames of the animation obtained during the ***IM*** calculation. Both the time required and the distribution of samples is identical.
--- DVD ANIMATION ---

From the test we have just carried out, we can see that both methods produce the same identical ***IM***. This is confirmed also by the size of the ***IM*** which in both cases is the same size.

■ **Figura 3.167**
IM size for the ***Multiframe Incremental*** method and the ***Incremental add to current map*** method.



The difference between the two systems is merely the way the ***IM*** is managed. With the ***Incremental add to current map***, it is possible to re-use a previously saved ***IM***. With the ***Multiframe*** method one cannot.

There is also a series of commands which are easily understandable which allow one to manage the ***IM***.

■ **Figura 3.168**
Screen with commands from the ***Mode*** panel.

Browse - this command lets one load the ***IM*** from a chosen path. One can also type the path directly into the text box.

Save to file - this command lets one save the ***IM*** stored in the memory after the conclusion of the last render. One must make sure that the ***Don't delete*** box is activated. Otherwise, *VRay* deletes the ***IM*** from the RAM at the end of each rendering session, making the attempt to save manually through the ***Save to file*** command completely useless.

Reset irradiance map - this command erases the ***IM*** permanently from the memory. Doing this is useless in ***Bucket mode***, because *VRay*, with this method, does not store ***IM*** information.

On render end

The last part of this complicated rollout concerns the ***On render end*** section. These parameters allow one to manage the ***IM*** at the end of the rendering process. Once the ***IM*** calculation is complete, the map can be automatically erased, preserved in the memory, saved in a file or automatically loaded in the ***From file*** module.

■ **Figura 3.169**
These options allow one to manage the ***IM*** when the render is finished.

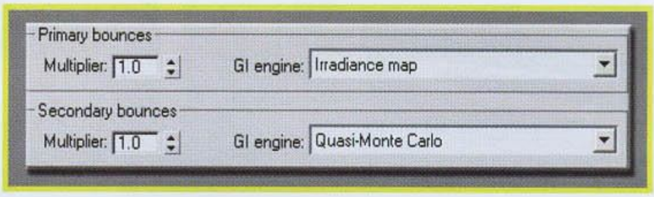
Don't delete - this is the default function. At the end of the ***IM*** calculation, *VRay* does not delete the map and all the memory-held data, but instead keeps it in standby for possible following operations, like normal or automatic save procedures. If the ***Don't delete*** parameter is disabled, at the end of the render the ***IM*** will be automatically erased.

Auto save - if this is active, one can input the destination path of the automatic save.

Switch to save map - this parameter is available only when ***Auto save*** is active. With it, once the render is over, *VRay* automatically loads the ***IM*** in ***From file*** mode.

TYPES OF USE

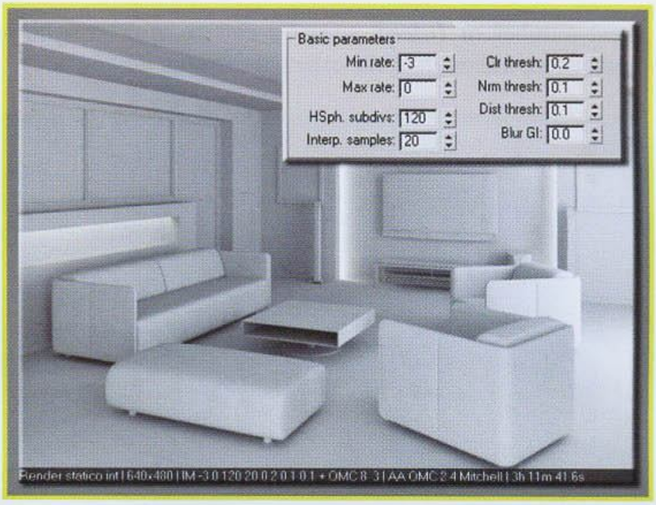
So far only two methods for calculating GI have been explored: ***QMC*** and the ***Irradiance Map***. Earlier we spoke of the types of scenes which can be rendered using only the ***QMC+QMC*** method, both or ***Primary*** and ***Secondary bounces***. Now we can add the ***Irradiance Map***. If we take a look at the dropdown menus ***Primary bounces*** and ***Secondary bounces***, it is to be noticed how the ***IM*** is available only for calculating ***Primary bounces***. Hence the only combination for GI calculation is, for now, ***IM+QMC***.



■ **Figura 3.170**
The ***IM*** Engine is selectable only in ***Primary bounces***.

. INDOOR STATIC RENDERS

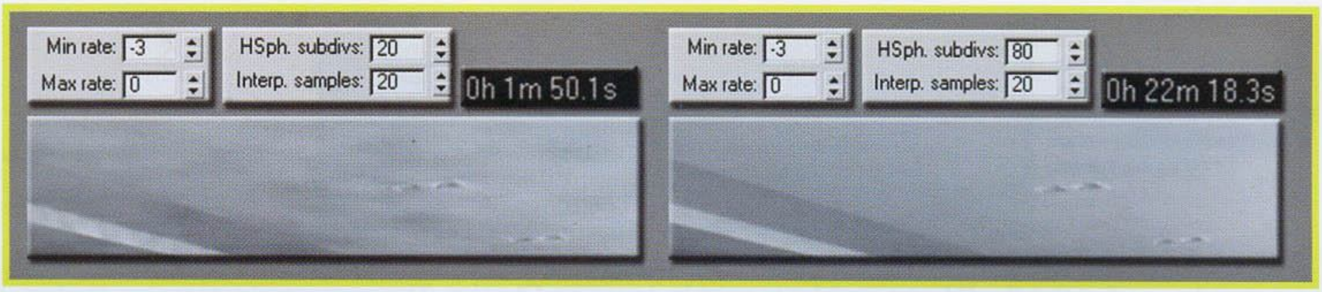
The ***IM+QMC*** system is definitely a more convenient method than the ***QMC+QMC*** method. The asset of approximation, and thus faster rendering, make the ***IM*** a top choice for static renders. As we will see later, the choice of ***QMC*** for ***Secondary bounces*** is not the best of choices, but is however a possibility and the results are certainly of excellent quality because of its unequaled precision.



■ **Figura 3.171**
This image was generated with ***IM*** and ***QMC***. The ***IM*** values are quite high, as well as ***HSph. subdivs***, whereas the ***QMC*** parameters are the default ones. The rendering time is high for the simple reason that ***QMC*** has been used for the ***Secondary bounces***.

By using the pair ***IM+QMC***, the following problems can arise:

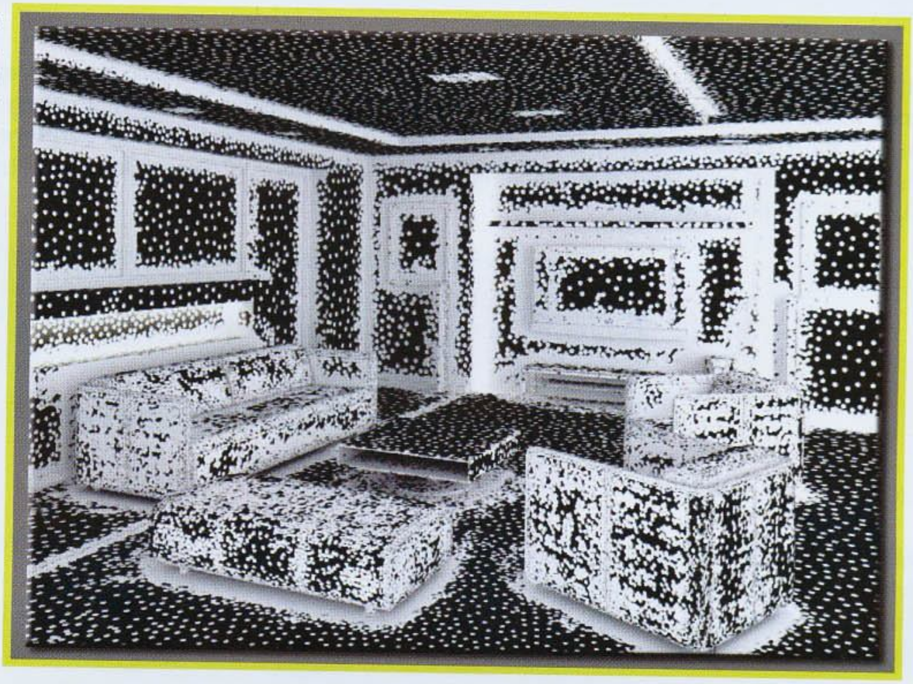
1. Scattered imperfections in the scene. The solution could be to increase the number of ***HSph. subdivs***. This would assure a satisfactory number of rays for the GI calculation.



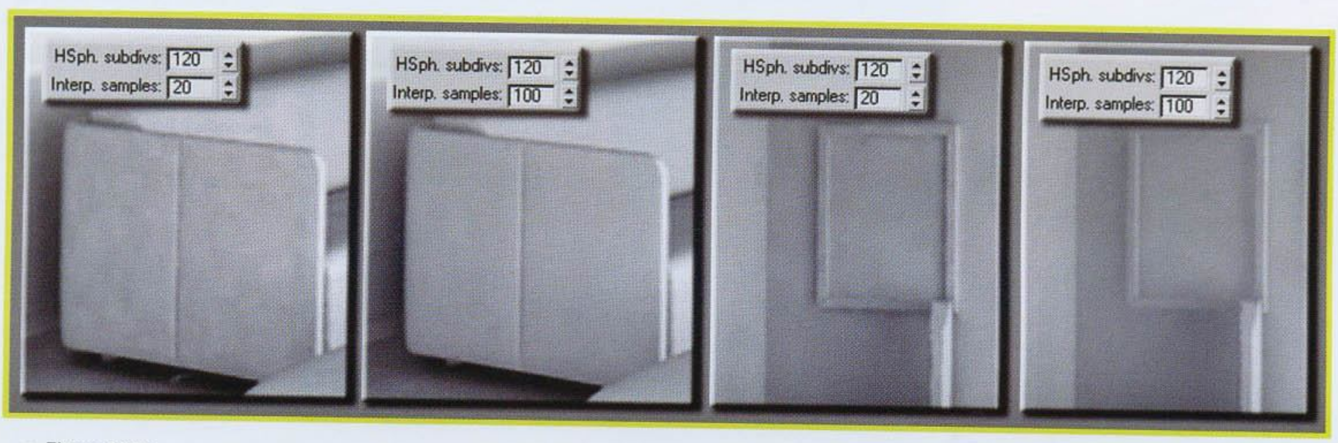
■ **Figura 3.172**
By increasing the number of ***HSph. subdivs*** also GI quality increases. The spots, which are a typical problem of the ***IM***, are eliminated in exchange for rendering time.

However, this is not the only solution. One could also consider increasing the ***Interp. samples*** parameter. This way the GI would be softer and the problem of the blotches would be lessened, but the image would be less detailed. In the example, high values have been used for calculating the ***Irradiance Map***. This has assured a high number of samples.

▪ **Figura 3.173**
Sample representation in the ***IM***. By using high ***IM*** values, an excellent distribution of the samples has been obtained. To be completely honest, in this example ***Clr threshold*** is so low that it has caused an unnecessary concentration of samples near curved surfaces.



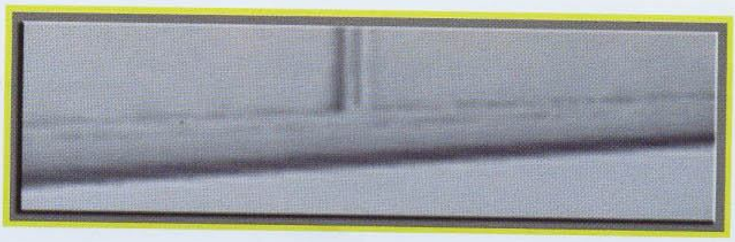
In this particular case, a great number of ***Interp. samples*** can be a valid solution for getting rid of imperfections, nevertheless allowing good precision, although at the expense of small details.



▪ **Figura 3.174**
The increase of the ***Interp. samples*** has a blending effect on the GI, as we can see on the side of the couch, but the side-effect can be a loss of detail, as can be seen in the bottom part of the picture frame. Usually one does not apply values beyond 50.

2. Another problem that could be faced is precision problems in very complex areas.

▪ **Figura 3.175**
Areas like frames and inlets are the main enemy of the ***IM***. Problems such as missing details or smudges are symptoms of an ***IM*** lacking in precision.



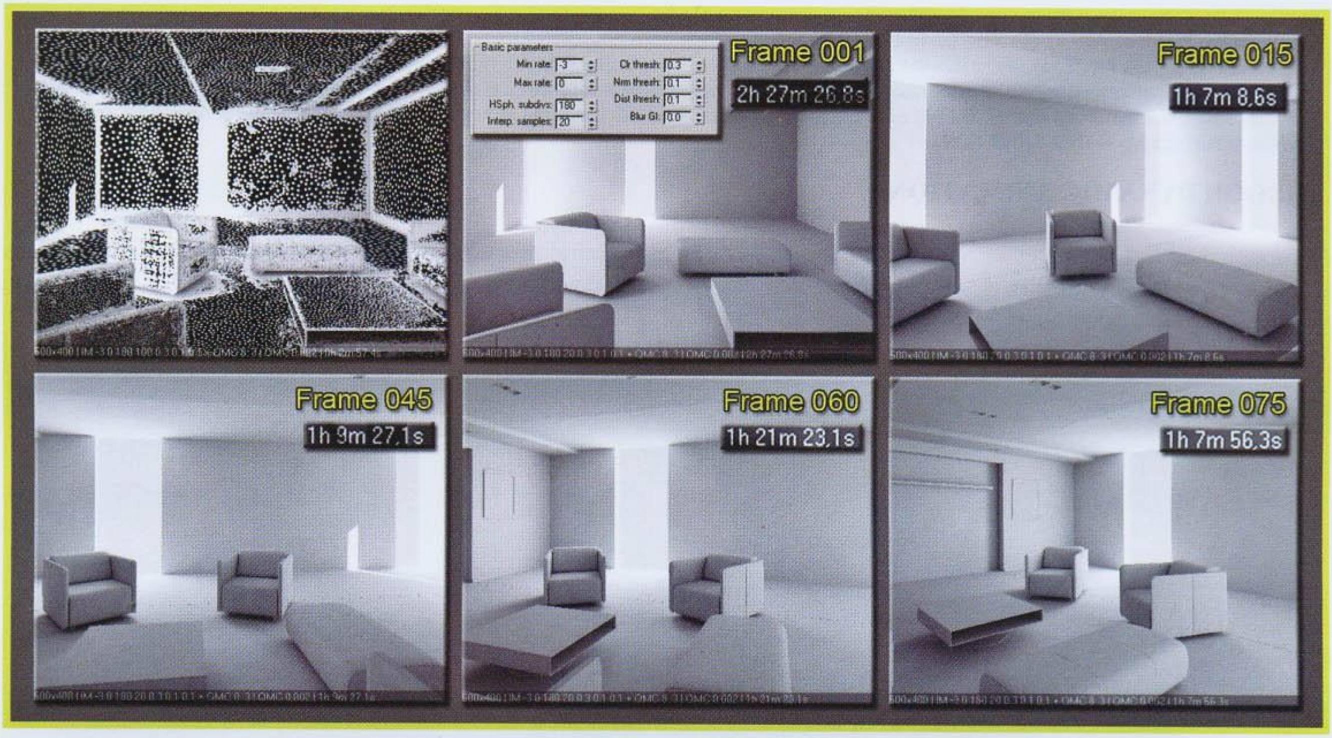
It is a well-known fact that the ***IM*** uses samples to quickly determine GI value. The more samples there are, the better GI quality is. In order to solve the problem of detail loss, it is more important that the samples be precise rather than numerous. Therefore one can increase the number of ***Prepasses*** of the ***IM***. The greater the ***Max rate*** value is, the more precise samples are.

. INDOOR STATIC ANIMATIONS (fly-through)

This is one of the fields where the ***IM*** excels. The fact of being able to save and re-use the GI makes the ***IM*** ideal for calculating ***Primary bounces*** for ***Fly-through*** animations. Not even advanced methods such as ***Photon Map*** and ***Light Cache*** can match its efficiency.

The first thing which must be done is to calculate the ***IM*** and save the ***.vrmap*** file. In this type of animation the only thing moving is the camera. Imagine walking into an office. At first one finds oneself in front of a new room, and a new ***IM*** must be calculated for it. Then, as one walks further into the building, new areas become available. Obviously for new geometries it will be necessary to calculate a new ***IM***, and add the new samples to the ones which have been calculated before. This happens with the ***Incremental add to current map*** method. Another way to save precious time is to take advantage of the fact that one can choose wether to calculate the ***IM*** every 5, 10 or 15 frames, according to how fast the animation is. If the animation, for example, proceeds at walking speed, in one second (roughly 24 frames) not much will happen. So to calculate the ***IM*** for each frame, although technically correct, is practically useless and expensive. Instead, the case of fast camera movements, if the GI calculation is not frequent enough, the risk can be to miss some samples. If this happens, one must add the missing frames and their samples. If not, the ***IM*** might not have the necessary samples for the correct calculation of those particular frames.

In the next exercise a camera has been placed in a room. The camera moves around the room in such a way that we get a 360° view of it.

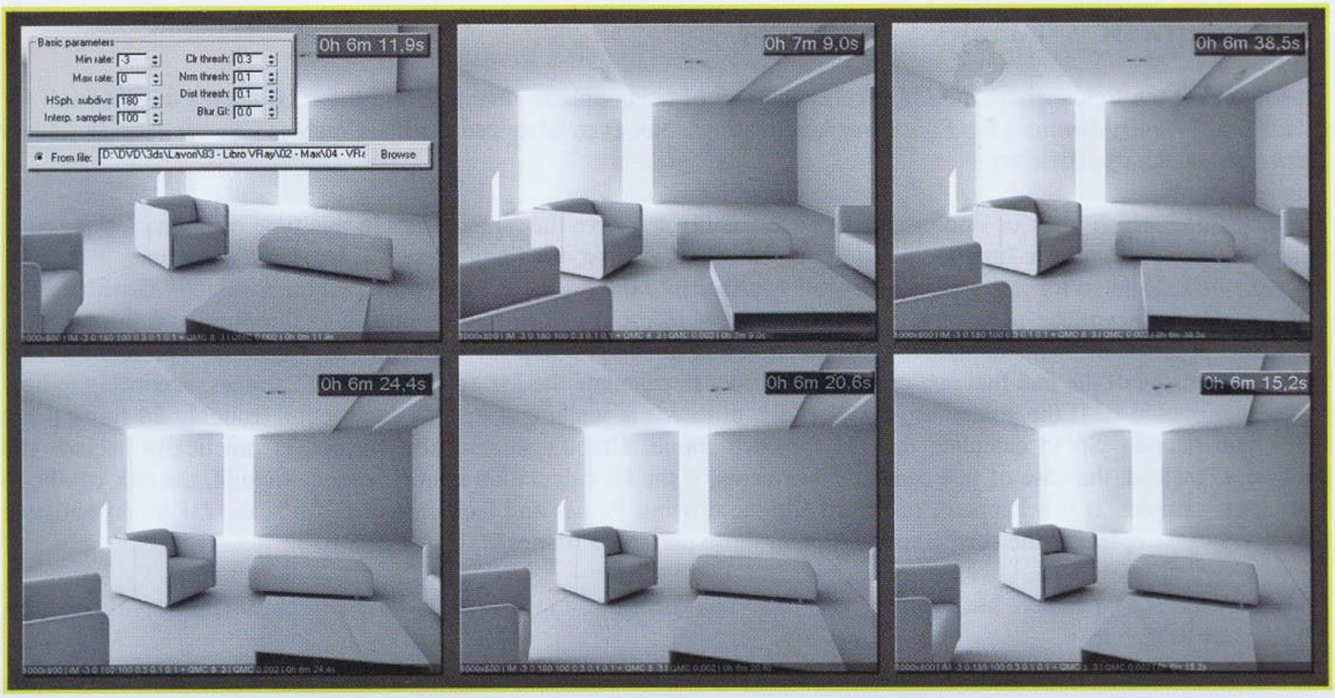


■ **Figura 3.176**

IM calculation of the first frame is the most costly in terms of time taken up. As this is the first frame, VRay knows nothing about the GI. Already during the calculation of the second render (in this example the 15th) the rendering time has suddenly diminished. In fact many ***IM*** samples have been calculated during the first render. In the second one, only the new samples of the couches, of the left-hand wall, new areas of the ceiling and the central table have been added.
--- DVD ANIMATION ---

A particular ***IM*** feature is the fact of being able to re-use it for rendering images at a resolution which is greater than the ***IM*** calculation itself.

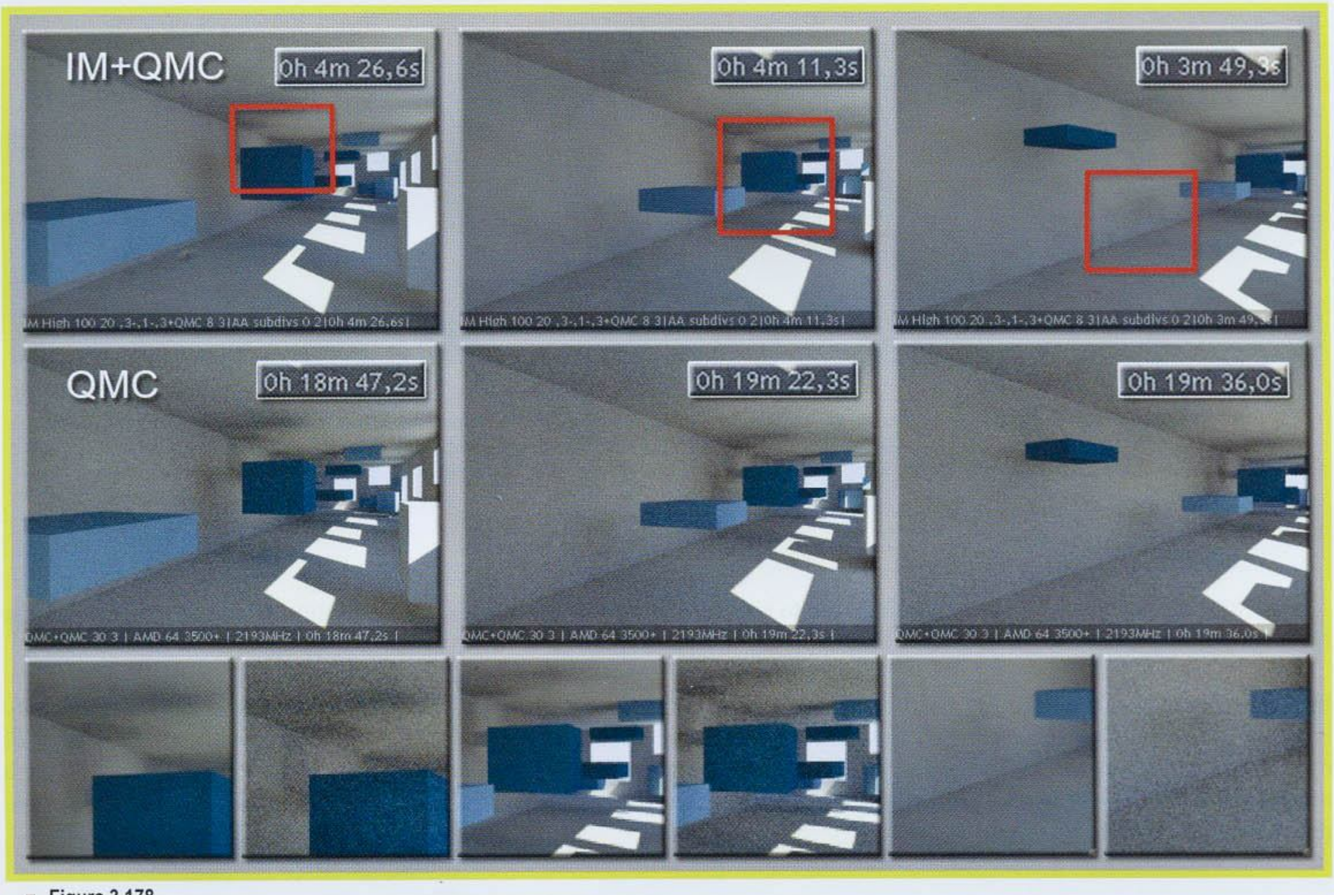
In the following exercise the ***IM*** has been calculated at a resolution of 500 x 400 pixels. The ***IM*** is very high quality because of the parameters used, and this allows it to be re-used for renders with twice the resolution. This type of operation always depends on the quality of the ***IM*** which has been calculated and it is not always reliable. If the ***IM*** at a resolution of 500 x 400 pixels had been calculated with low values and one wanted to re-use it for higher resolutions, problems and flaws could occur.



■ **Figura 3.177**
In the following exercise the ***IM*** has been calculated at a resolution of 500 x 400 pixels. The ***IM*** is very high quality because of the parameters used, and this allows it to be re-used for renders with twice the resolution. This type of operation always depends on the quality of the ***IM*** which has been calculated and it is not always reliable. If the ***IM*** at a resolution of 500 x 400 pixels had been calculated with low values and one wanted to re-use it for higher resolutions, problems and flaws could occur. - - - DVD ANIMATION - - -

. INDOOR DYNAMIC ANIMATIONS

Chromatic and light changes or moving objects require *IM* calculation every frame. Therefore the benefit that *IM* gives in being re-usable is, in this case, of no help. It is to be remembered that *IM* is quick to be calculated and this can drastically reduce calculation time per frame.



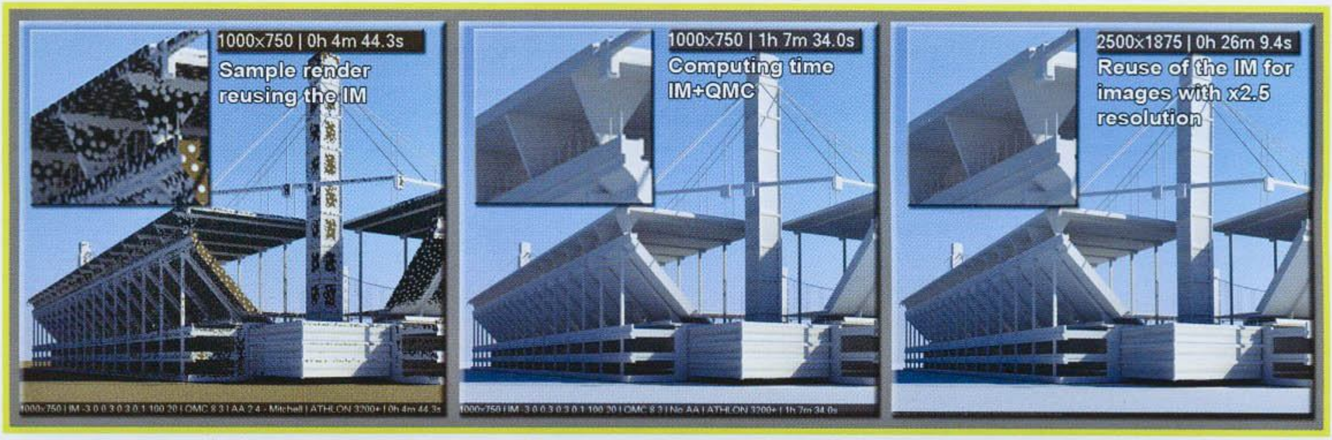
■ **Figura 3.178**
Comparison between the ***QM+QMC*** method and the ***IM+QMC*** hybrid. The time required is much less and noise is lower. In this example low ***IM*** values were used, in order to emphasize speed, cleanliness and unfortunately, loss of details and also in order to reproduce the problem of flickering.
- - - DVD ANIMATION - -

In dynamic animations one of the main problems of the ***IM*** is flickering, which, as the name suggests is the flickering of colors and shadows between one frame and the next. But what causes this phenomenon?
Imagine rendering a scene with low ***IM*** values. The render would obviously contain defects, blotches and imperfections, but a static image could probably be satisfactory. Now picture a series of frames, for instance an animation leading through some rooms, with the same low quality. As for the single render, each frame will have imperfections because of the low number of samples. Because of this, for each frame the ***IM*** is slightly different. This means that in a certain point of the model, the ***IM*** could change value slightly: in other words it could be slightly lighter or darker. If you now put these frames into a running sequence, you will be able to notice how the animation has slight variations in color and lighting due to the low quality ***IM***. The only way to solve this problem is to use higher ***IM*** values, especially in the ***HSph. subdivs*** parameter.

There are cases where the use of ***IM*** for dynamic animations could not be the best solution. When there are ***glossy*** shaders, ***Area Shadows***, translucence, ***Motion blur*** and ***DOF*** in the scene, it may well be that the use of ***IM*** is not such an advantage in terms of time/quality compared to the use of ***QMC+QMC***. If one uses this last method however, one must be sure to possess ample hardware.

. OUTDOOR STATIC

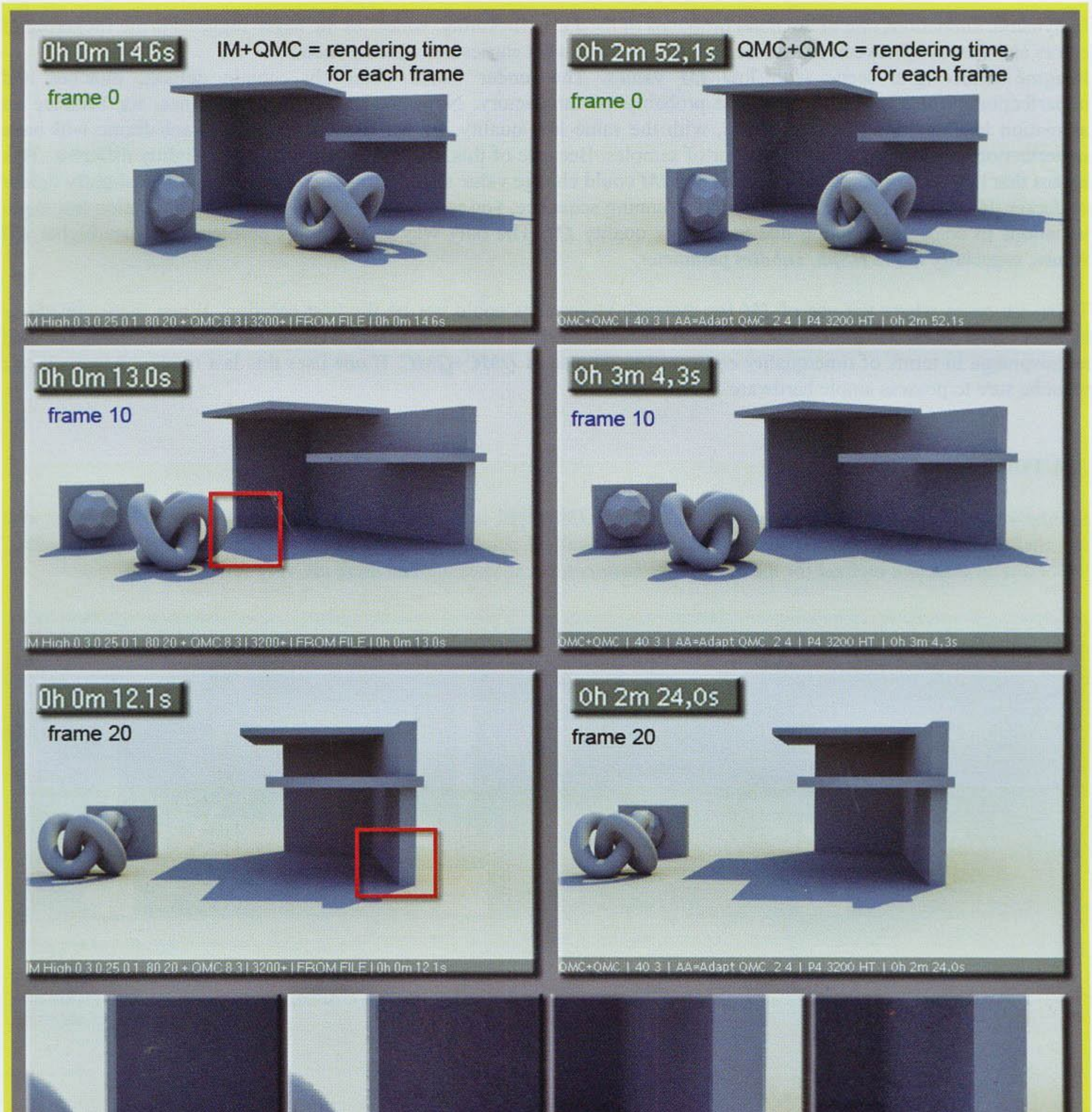
Similarly to static indoor renders, also for outdoor use, ***IM+QMC*** is the best choice. The speed of the ***IM***, together with the quality of the ***QMC*** method makes this pair an excellent solution for high quality renders. In this case the ***QMC*** method is an effective method for the ***Secondary bounces***, because of the fact there are very few.



■ **Figura 3.179**
Here the ***Irradiance Map*** has been calculated with middle-high values. This has enabled the image to be rendered at a resolution of two and a half times greater, achieving first-rate results.

. OUTDOOR STATIC ANIMATIONS (fly-through)

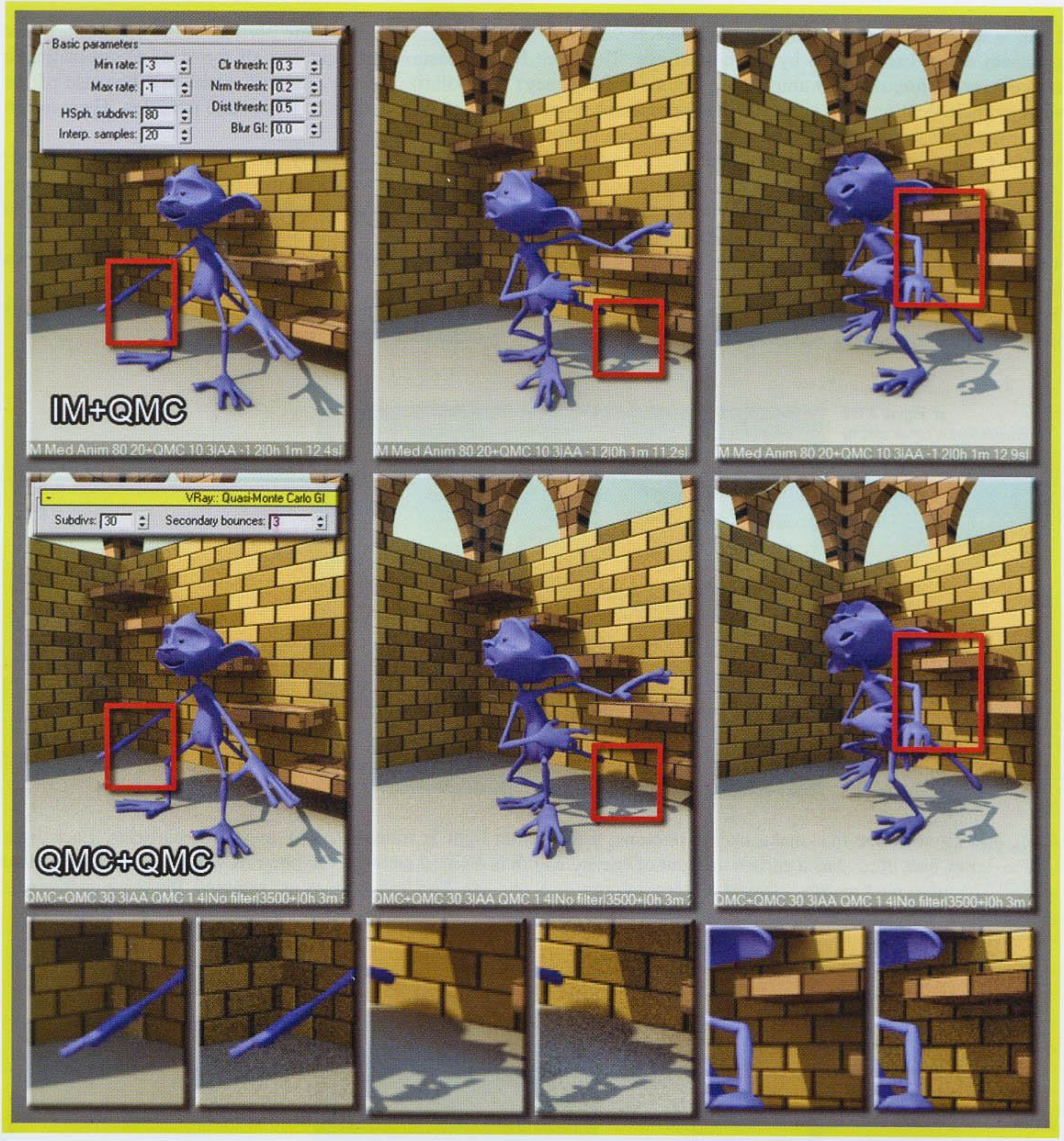
The ***Irradiance Map*** is definitely the best choice also for this type of scene. The ability to save and re-use the GI makes the ***IM*** a winning tool. The ***QMC*** for ***Secondary bounces*** also guarantees the quality of the final animation.
In the next example the ***QMC+QMC*** method and the ***IM+QMC*** method are compared. For ***IM+QMC*** the first step is the calculation of the ***IM***. The animation can be rendered every "n" frames, according to animation speed. This already saves a lot of time. Otherwise the ***IM*** can be calculated for every frame. For this animation, which is 100 frames long, the average for ***IM*** calculation is around 30 sec. per frame. One has chosen to calculate the ***IM*** every 10 frames, therefore obtaining a total calculation span of 300 sec., which means 3 sec./frame (300 sec./100 frames total). Add 13 seconds to each frame for final rendering and you have a total time of **16 sec. per frame**, as opposed to **2 min. 30 sec.** roughly for the ***QMC+QMC*** method. As one can see, also the quality has suffered very little.
It can be safely said that the choice of ***IM+QMC*** is by far a better choice than the ***QMC+QMC*** method. Much like in other examples, however, when ***Area Shadows***, ***glossy*** or translucence effects, ***SSS***, ***Motion blur*** or ***Depth of field*** are present, using the ***IM*** can be a bad idea. In this case, the only thing to be done is to test the scene with different methods and see which returns the best results. Unfortunately there is no hard and fast rule in this case.



■ **Figura 3.180**
Comparison between the ***QMC+QMC*** method and the ***IM+QMC*** method. With this last method, timespans and noise are very low.
- - - DVD ANIMATION - - -

. OUTDOOR DYNAMIC ANIMATIONS

For outdoor situations, GI calculation is generally faster compared to indoor ones because of the fact that very few bounces are needed before the rays of light consume their energy, no matter what method is used. It is therefore possible to consider using ***QMC+QMC*** as opposed to ***IM+QMC*** for GI calculation, even though the latter can guarantee less than half the time needed. From the point of view of quality, when possible, it is always best to use ***QMC+QMC***. Scenes with outdoor dynamic animations are cases where this method could be efficiently applied



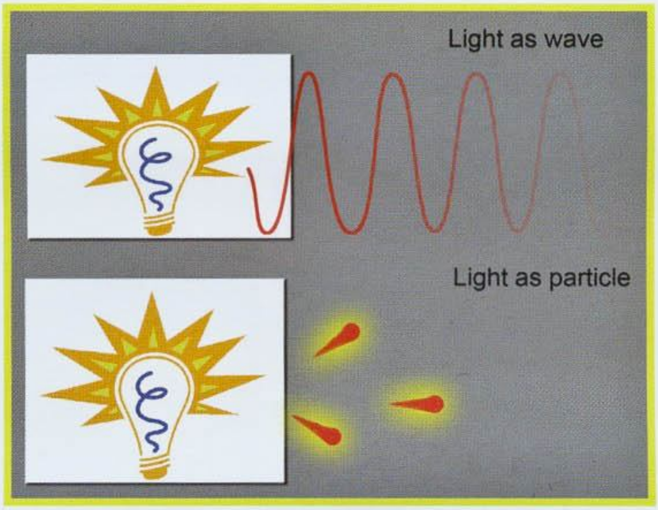
■ **Figura 3.181**
In this example the ***IM+QMC*** method is compared with the ***QMC+QMC*** method. One can notice the presence of noise with the ***QMC+QMC*** method. As far as calculation duration is concerned, the ***IM+QMC*** method only requires half the time, lowering it from 3 min. to 1 min.
--- DVD ANIMATION ---

VRay: Photon Map(PM)

INTRODUCTION

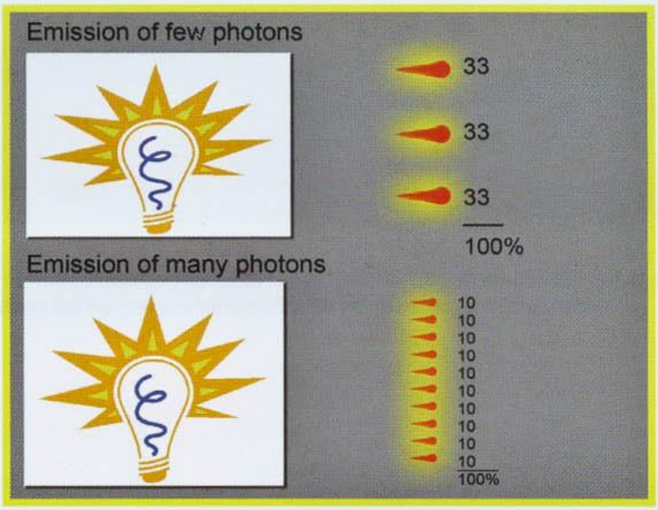
Light is the physical agent which makes objects visible. The term "light" refers to the portion of the electromagnetic spectrum which can be seen by the human eye. The three basic measures for light and all forms of electromagnetic radiation are luminosity (or amplitude), color (or frequency) and polarization (or angle of vibration). Because of the wave-particle duality, light simultaneously possesses properties which belong both to waves and particles. There are in fact two theories supporting the physical definition of light.
The first theory goes by the name of ondulatory, and was formulated by Christiaan Huygens in 1678 and published later in 1690 in "***Traite de Lumiere***". In this composition, light is regarded as a wave which moves in exactly the same way as the waves in the ocean or acoustic waves which were thought to travel in medium called ether, which was believed to be all over the universe and made of microscopic elastic particles.
The second theory, formulated by Isaac Newton in the XVII century, established light to be made up of small particles of matter, small bodies called photons, which are emitted in all directions.
There is also a third theory, elaborated by James Clerk Maxwell, at the end of the XIX century. It sustains that luminous waves are electromagnetic waves and do not require a medium in order to be transmitted.

■ **Figura 3.182**
Physics light is considered by means of two systems. Light denoted as waves or particles called photons.

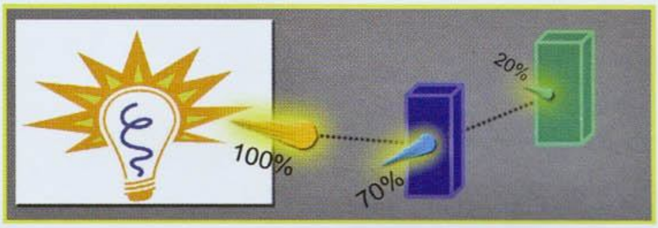


VRay and all software that make use of photons, assume that light is transmitted by means of particles. Each light source, lamp, sun, etc… has a certain amount of energy which is divided amongst the photons which are emitted by that source. The more photons are released, the less energy each one has. The less photons are released, the more each one is charged with energy.

■ **Figura 3.183**
The total energy emitted by the light source is divided into equal parts for each photon. The more photons are emitted, the less energy they contain.

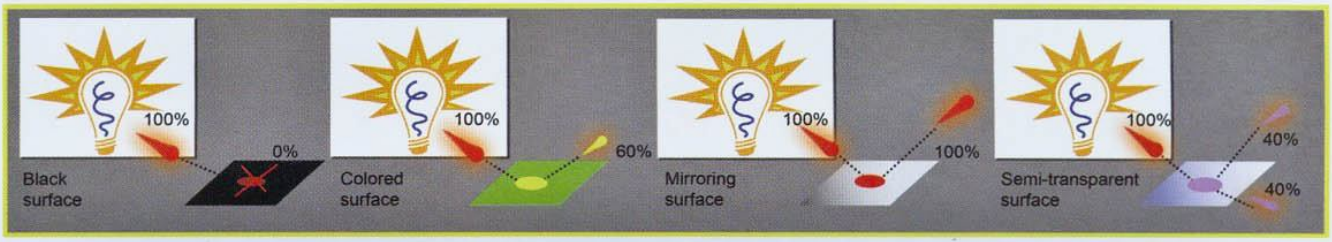


Photons have two fundamental characteristics: color and energy. Initially their color is defined by the light source, likewise their energy is defined by the value given to them by the light source. After colliding with objects, photons absorb a portion of color belonging to the object they have encountered, and yield a certain amount of energy. The more collisions there are, the more energy will be lost.



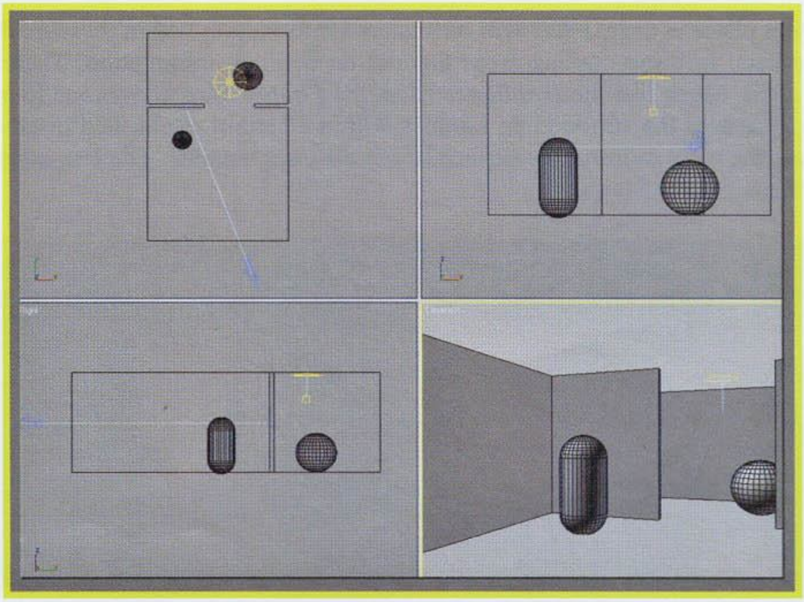
■ **Figura 3.184**
If photons do not collide with a surface which is completely absorbing, they bounce off, each time losing a certain amount of energy.

The calculation of photons takes place before the actual rendering process. When a photon is born and hits an object, *VRay* analyzes the shader of the material of that object. According to the properties of that material, it decides to modify the photon, altering its color and energy.
There are two different behaviours that the photon can undergo after hitting a surface. The first possible case is that color and energy of the photon are completely absorbed by the material. The second case is that the photon is accordingly modified by the shader before moving on. It all depends on the object which has been hit.
If, for example, a photon hits a completely black surface, all light and energy would be absorbed. A colored surface, instead, would make the photon bounce off, changing both the color and the initial energy. A mirror would reflect the photon completely, without altering its energetic or chromatic characteristics. Semi-transparent surfaces allow a certain amount of the photon to pass through, whilst reflecting the remaining part.

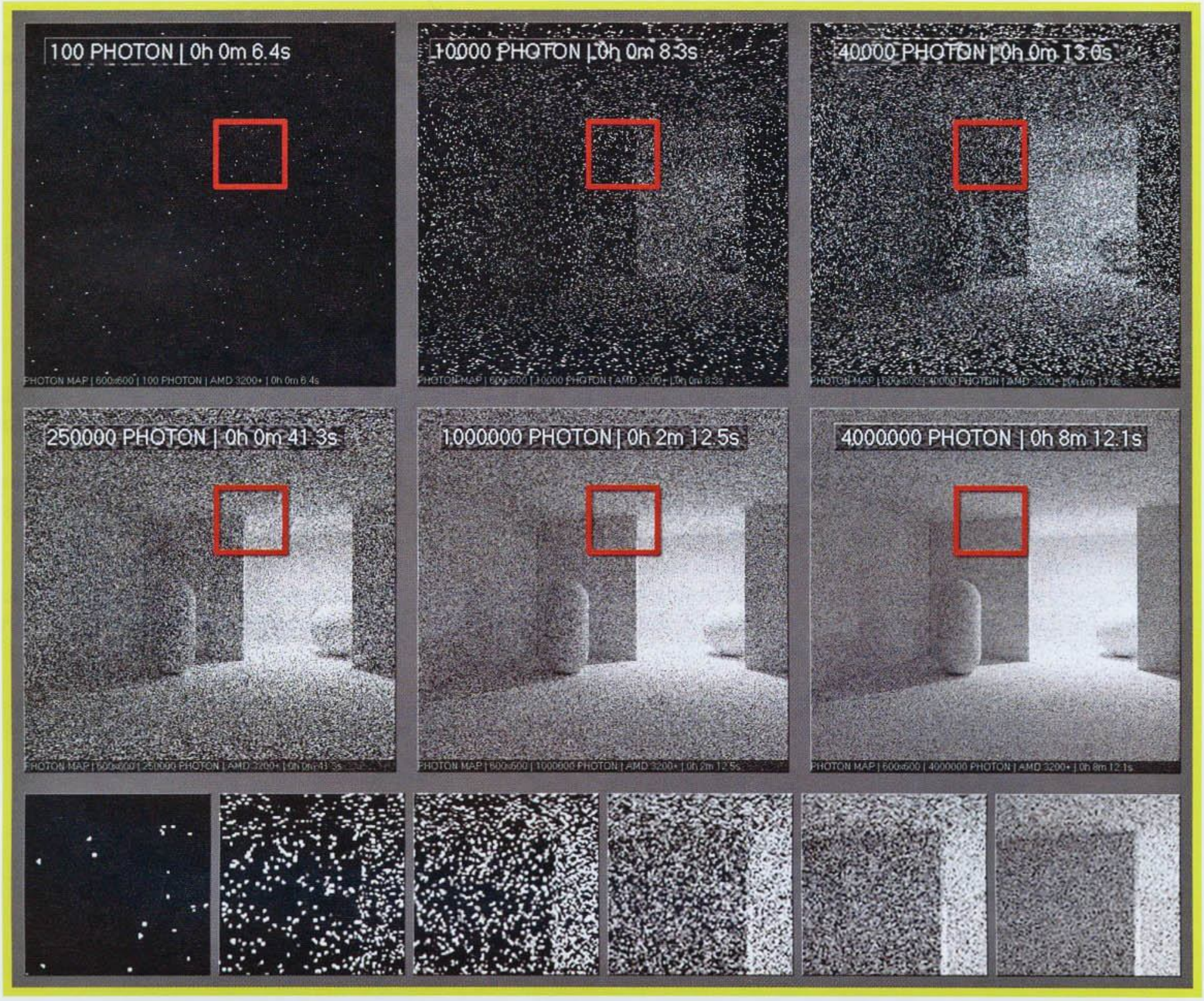


■ **Figura 3.185**
The behaviour of photons depends on the surface they hit.

The photons that hit a surface leave reference points called markers, which are similar to samples in the ***Irradiance Map***. The information in these markers is gathered in a 3d database called a ***Photon Map*** (***PM***). The more photons are used to calculate it, the more complex this map is. Also, the more photons that are used, the more demand there will be for system resources, time, memory and hardware for calculating the ***PM***.

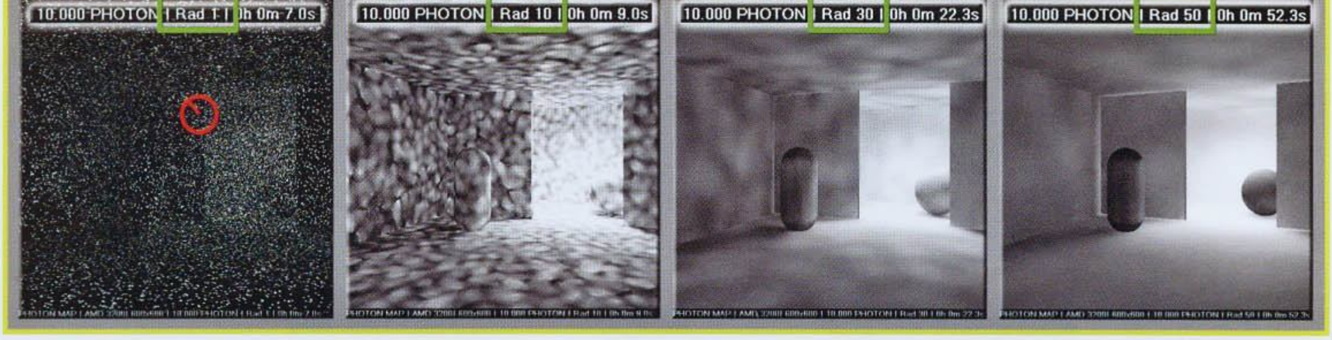


■ **Figura 3.186**
Screenshot of 3ds Max. Orthogonal views and model perspective for the next examples. There is only one light source.



■ **Figura 3.187**
The more photons there are, the better the ***PM*** quality will be, thus reducing noise. It is obvious how rendering time increases exponentially as the photons increase in number. Also notice how the photons are more luminous, the less there are of them. In the last image one can see what is one of the main problems of the ***Photon Map***. Edges are much darker, almost outlined in black. Later on in the book, how to face this problem will be explained.

Once the marker calculation phase is over, *VRay* will have stored information within these markers. But not all of the surface is covered by markers. In these missing areas there is no defining information. Therefore it is necessary to define the value of these areas. Much like the ***Irradiance Map***, the ***Photon Map*** allows one to interpolate the markers. In areas without samples, one uses the surrounding samples within a certain established distance to calculate the GI. The more markers are used, the more uniform the surface is, but also less detailed. The less samples are used, the more detailed the render is, but there may be blotches.



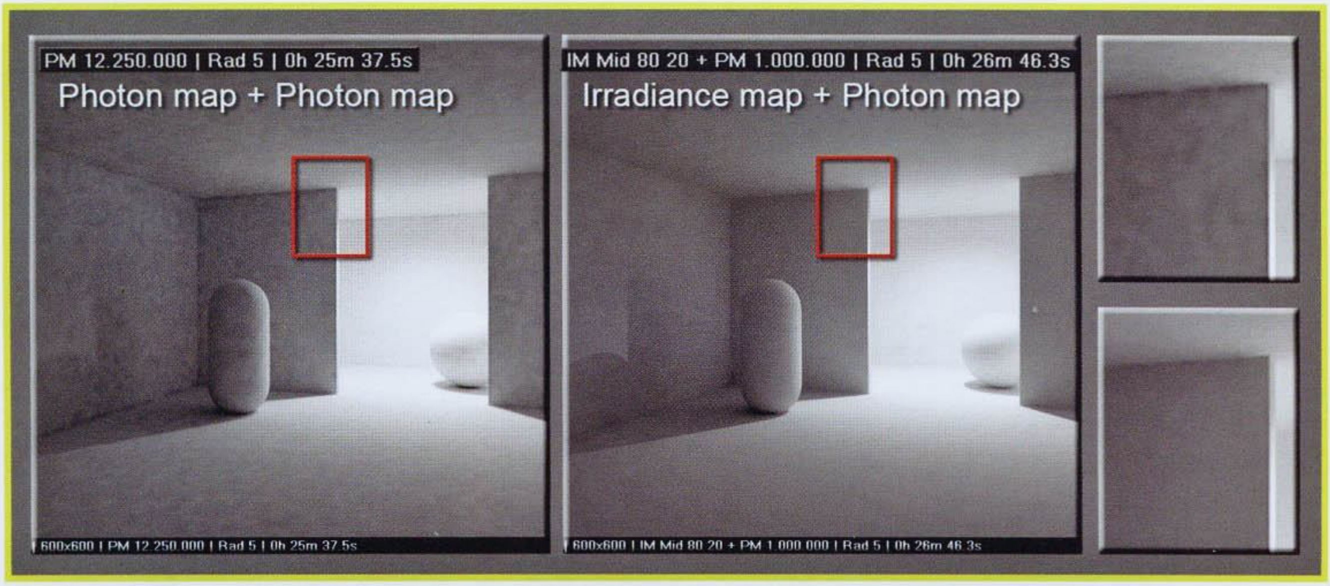
■ **Figura 3.188**
The red area defines the circular interpolation radius. The larger the interpolation surface is, the more uniform the render is, but less detailed.

For the sake of convenience, *VRay* separates photons into two types according to their purpose and use: the first type is responsible for the calculation of the Global Illumination, whereas the second type is used for calculating caustic effects. Caustics are a phenomenon which takes place in nature when beams of light converge on the same surface. They can be observed in real life and can be generated by reflection (rings, metal objects, etc.) or refraction (through water, liquids or transparent materials).

■ Figura 3.189
Caustic effects in nature.

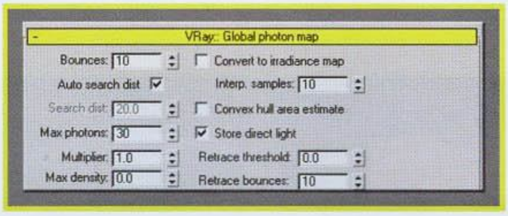
Usually, for the calculation of caustics, a much smaller interpolation radius is used compared to GI calculation. This is because of the intrinsic nature of caustics, which are concentrated effects with a lot of details.

One of the main problems when using photons is deciding their number at the beginning, in order to fulfill the scene well, without overloading rendering times. In the previous examples, the number of photons was high, around 4 million, and the scene was still rough and grainy. This problem is solved with a little help from the ***Irradiance Map***. Later on, we will explain how to use it together with the ***Photon Map***, but we can already anticipate that thanks to the ***IM***, we can greatly reduce the number of photons used compared to pure ***Photon Map*** application, both in ***Primary*** and ***Secondary bounces***, and nevertheless obtain a cleaner and more uniform result, in the same amount of time.

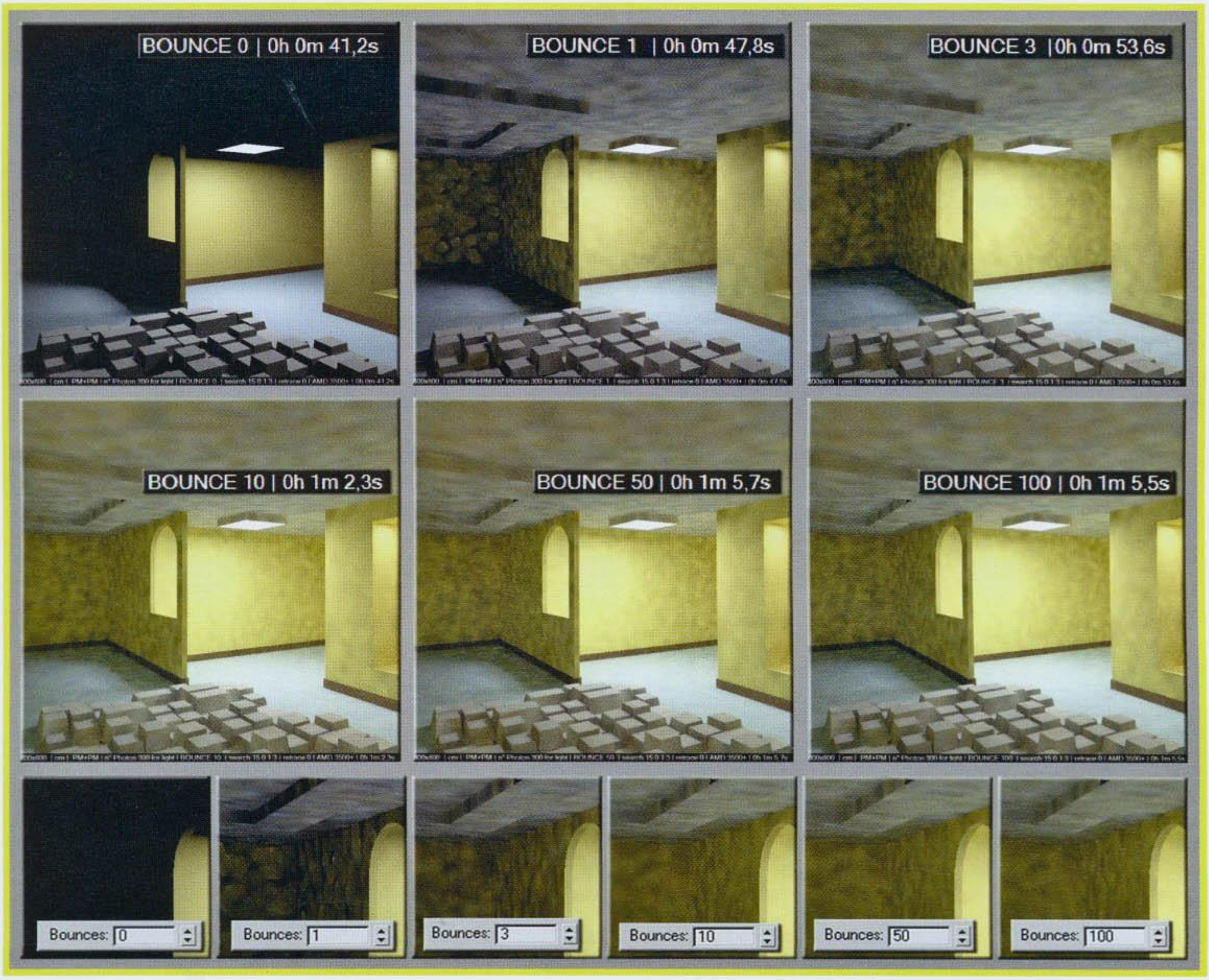


■ Figura 3.190
Comparison between the ***PM+PM*** method and the ***IM+PM*** method. In the same amount of time, with the ***IM+PM*** system, we obtain a much cleaner render. One can also see how problems can occur in the corners with ***PM+PM*** method.

■ Figura 3.191
VRay: Global photon map rollout.

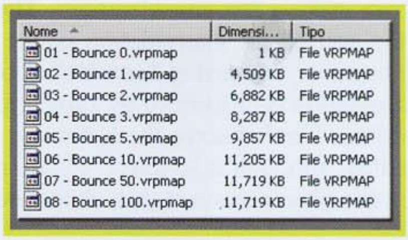


Bounces - this parameter controls the number of bounces that photons carry out during the calculation of the ***Photon Map***. The more bounces, the better the quality and lighting of the render, at the expense of memory and time.



■ Figura 3.192
Indoor scene rendered with the ***PM+PM*** technique. As the number of bounces increases, so does the GI quality in the areas with indirect illumination.

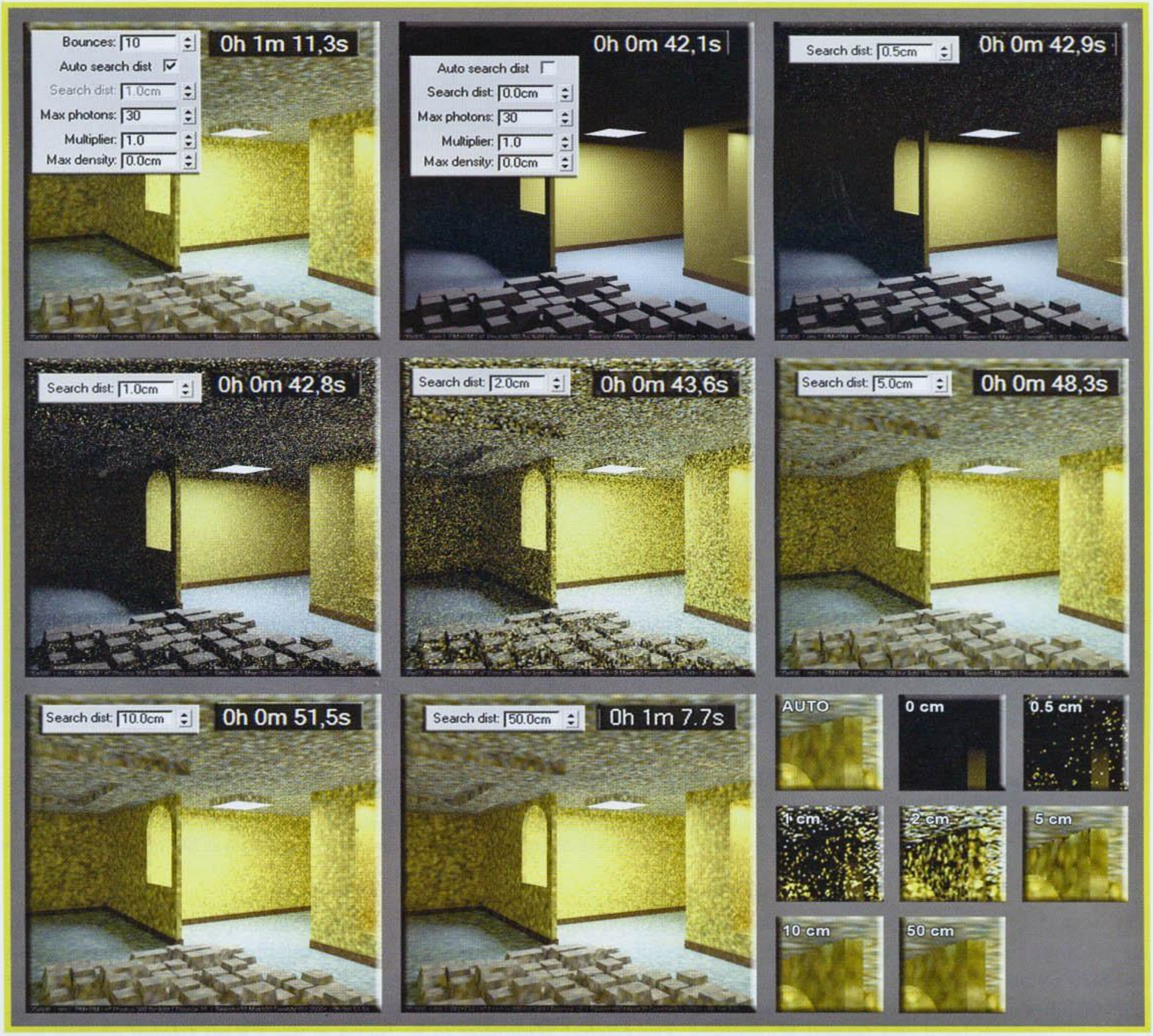
By using a bounce value of 0, GI is not computed, and the areas which are not directly illuminated will remain dark. This result could seem wrong. GI is activated, ***PM*** is being used but Global Illumination is completely absent. But it is correct instead, because *VRay* has been told to bounce photons once only. The particle is emitted until it collides with the first object, which it illuminates, but then its lifespan ends, as it cannot bounce again. With the parameter set to 1, the photons can collide with the first object, bounce, and illuminate areas indirectly. Set to 3, the lighting in indirect areas is greatly improved, and this is true until value 10. Beyond 10, increasing bounces does not improve the lighting in any way. The fact that it would be useless is confirmed by the size of the ***PM*** itself.



■ **Figura 3.193**
When using 10 bounces the size of the ***PM*** is almost 3 times bigger than when using a single bounce.

Auto search dist - when this parameter is active, ***VRay*** tries to automatically search for the right distance for photon interpolation. Sometimes this function can return satisfactory results. Sometimes the value is too great and can produce excessive render times. If it is too small, renders may have blotchy areas.
As we anticipated in the introduction, photons create a cloud of 3d points which contain information on the GI. But not all of the scene is covered by the markers and in the empty areas the GI is obtained via interpolation with the closest samples. With the ***Auto search dist*** function, ***VRay*** automatically attempts to find the best interpolation distance. By deactivating this function, the user will then define the value manually.

Search dist - this parameter is available when ***Auto search dist*** is disabled. It allows one to manually establish the interpolation distance for photons. The higher the value is set, the vaster the calculation area will be. Therefore there will be a greater number of photons used. This way the render is more uniform but also less detailed. Moreover, rendering calculation is slower. Small values make the render faster and more detailed but also prone to artifacts.



■ **Figura 3.194**
A render with various different ***Search distance*** values. Once the ***PM*** has been calculated, one can re-use the file in order to find the correct interpolation value, which saves a lot of time.

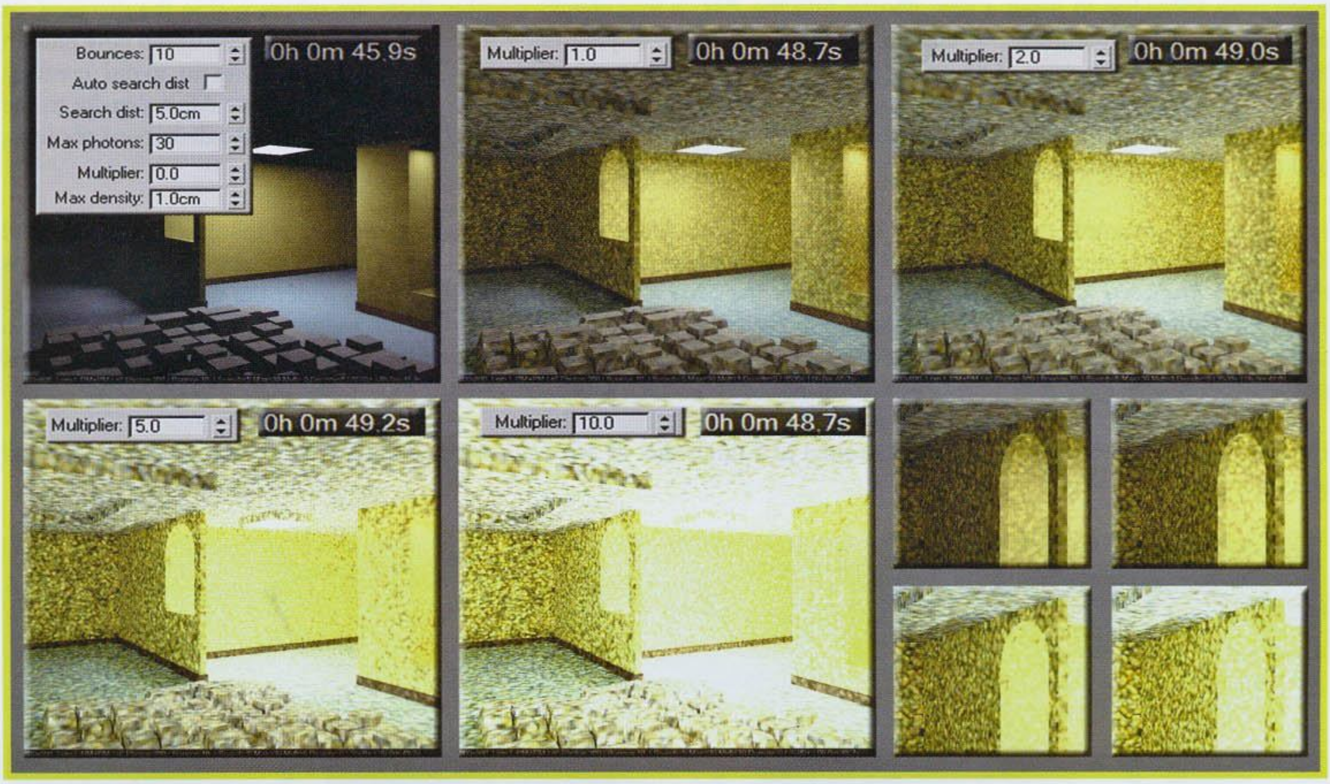
Thanks to these tests one can make a few considerations. The greater the ***Search distance*** parameter is, the longer the render takes. The vaster the surface for interpolation is, the more photons are used for calculating GI interpolation. Above a certain threshold, however, increasing ***Search dist*** does not bring about any change whatsoever. The reason for this is that the area for interpolation is influenced by another parameter: ***Max photons***. Soon we will see how these two controls depend on each other. Now observe the behaviour of the render with ***Search dist*** = 0. In this case indirect illumination is not represented, although it is active. Please remember we are using a ***PM*** which has been calculated previously. The reason for this phenomenon is that, as the area to be interpolated is equal to zero, there can not be any photons contained in it.

Max photons - this value defines the maximum number of photons to be used for interpolation. The more are used, the softer the image is, at the expense of time and detail. ***Max photons*** can be seen as a limiter. If *VRay* tries to use more photons than the amount set in the Max photos parameter for calculating the GI interpolation, it will be blocked as it would be going above the threshold that the parameter allows. Changing the ***Max photons*** parameter does not require a repeated calculation of the ***Photon Map***; a previously calculated one can be used.

Figura 3.195

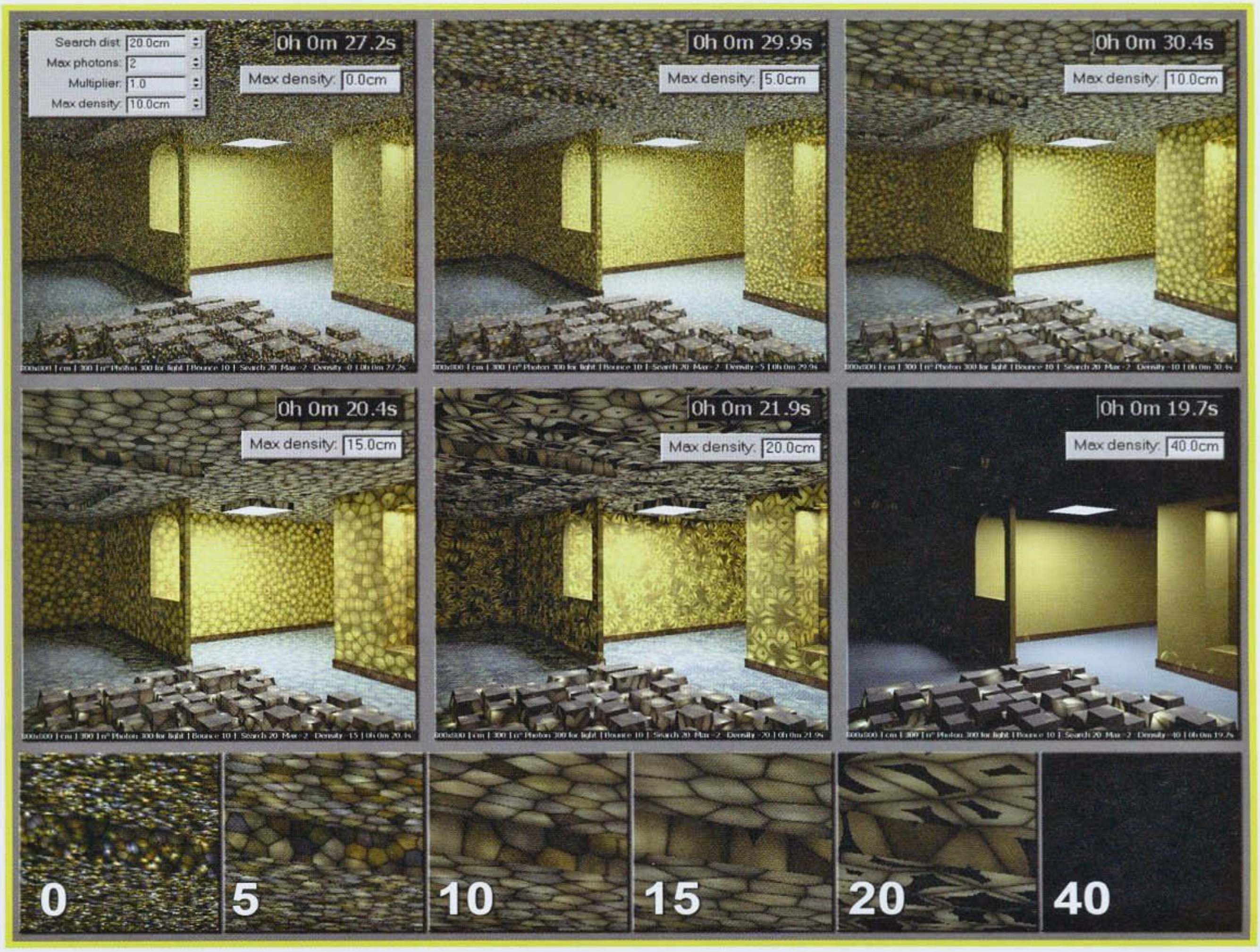
In the first two examples, with ***Max photons*** reset n observe the influence of ***Search dist*** alone. If one uses very low ***Search dist*** values, one can make out the single photons. As the value of ***Search dist*** is increased, the photons start to bend together, on the exclusive basis of their radius of influence. By introducing ***Max photons***, in other words moving it to values greater than 0, one can accurately define the number of photons to use for interpolation. Interpolation is not possible with one photon only, as we an see from the pitch black render. With two them, the ***PM*** is very similar to an ***IM***. The optimal values are between 10 and 30 photons.

Multiplier - this parameter controls the intensity of photons. It is a global value which multiplies luminous intensity for all of the present photons in the scene. As we will see later on, with *VRay* one can establish the number of photons and their intensity for every single light source. ***Multiplier*** instead, acts on the global intensity of all photons. ***Multiplier*** is cumulative, meaning that it is summed to the independent ***Multiplier*** of each source. There is a small difference, however, between local and ***Global Multipliers***. The ***Global Multiplier*** to be found in the ***VRay: Global photon map*** rollout can be applied without having to recalculate the ***Photon Map***, whereas changing the local ***Multiplier*** requires the map to be recalculated again.



■ **Figura 3.196**
The ***multiplier*** amplifies the global intensity of photons.

Max density - allows one to limit the resolution of the ***PM***, and therefore the amount of memory taken up by it.

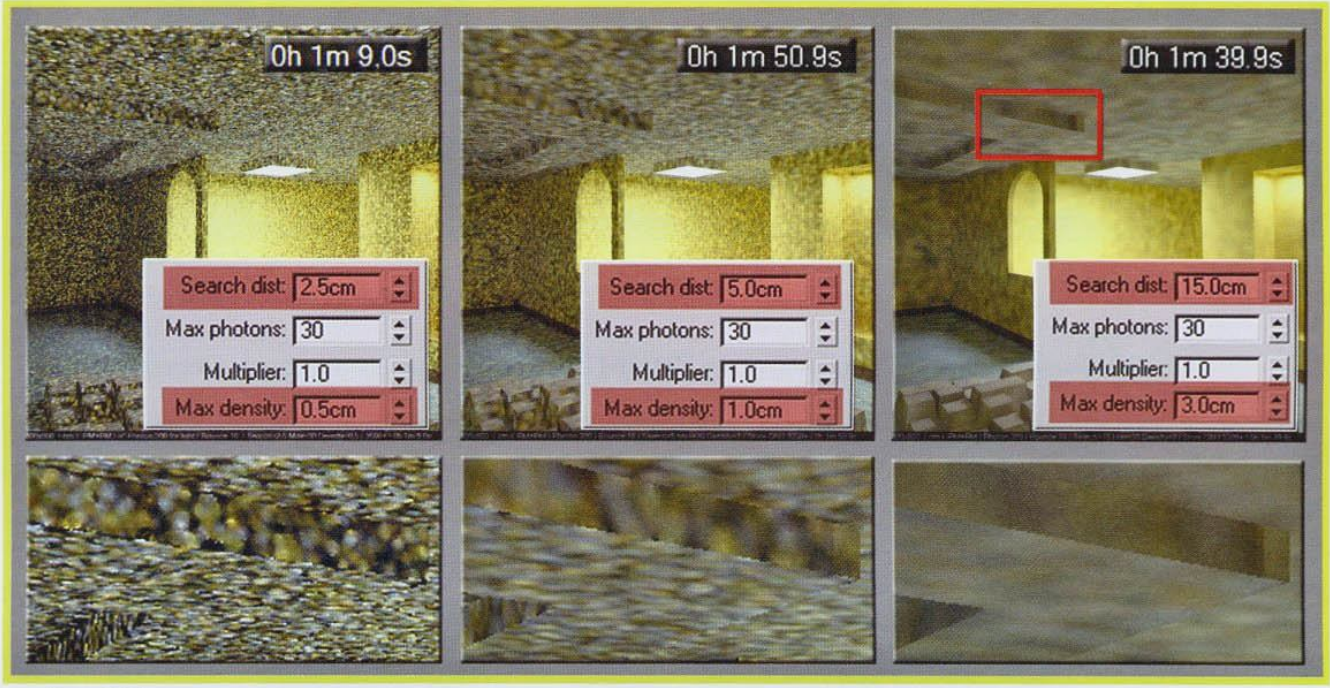


■ **Figura 3.197**
At parity rate ***Search dist***, the more ***Max density*** is increased, the lower the ***Photon Map*** density sinks.

Each time ***VRay*** has to insert a new photon into the ***PM***, it first checks whether there are others within the distance set by the ***Max density*** parameter. If there is a sufficient number of photons, ***VRay*** distributes the energy of the new photon amongst the existing ones. Otherwise it will simply add the photon. This method allows one to avoid creating excessively big ***.vrpmap*** files, whilst guaranteeing a correct number of photons. One can compare ***Max density*** to the mesh subdivision function referred to the ***Radiosity*** method. The smaller this value is, the greater the density of photons, and thus, a more detailed render.

In the previous example one can notice how, with a fixed ***Search dist*** value, as ***Max density*** is increased, photons tend to create an ever wider grid, until they disappear altogether. This behaviour is correct. If one thinks of ***Max density*** as the size of ***Radiosity*** subdivisions, then the higher the ***Max density*** parameter is set, the wider the geometric grids are. From this point of view, the ***Search Dist*** parameter (which regulates the interpolation radius of photons) is lower than the size of the meshes, there will be nothing to interpolate. And this results in a pitch black render. Thanks to ***Max photons*** being = 2, the zone of influence of the photons is well visible.

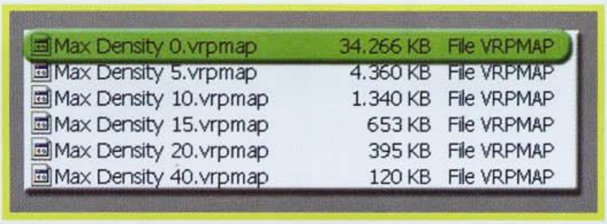
From experience, a value of ***Search dist***, from 3 to 5 times that of ***Max density***, guarantees a correct ratio between detail and uniformity of the GI.

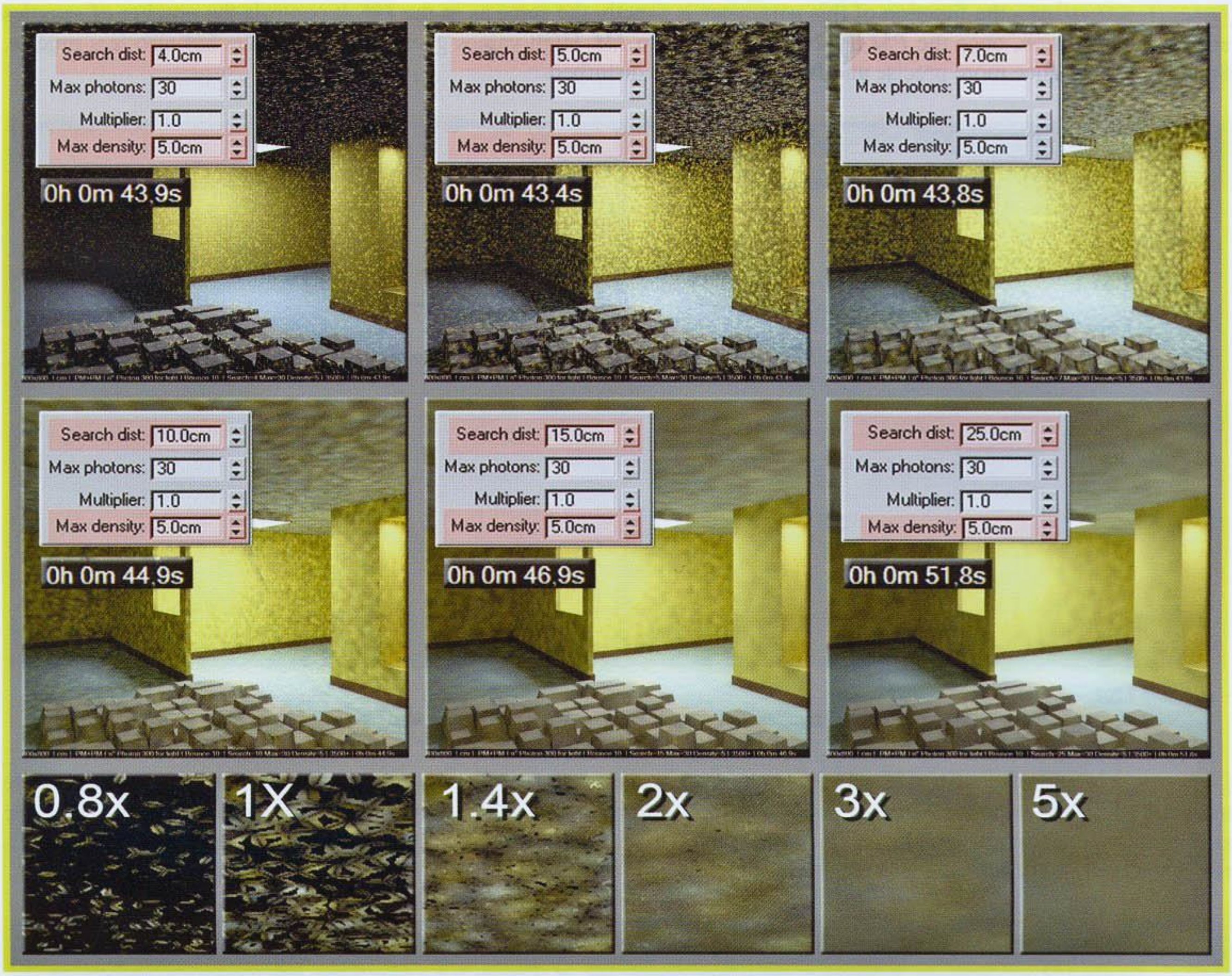


■ **Figura 3.198**
Search dist and ***Max density*** parameters are, in this example, always is a 1/5 ratio. The main difference in the tests lies in the size of the photon grids, defined by the ***Max density*** parameter.

The size of the ***.vrmap*** shows that with ***Max density*** = 0, the map is much heavier. In fact it is possible to see the area of influence of each single photon. By increasing the setting, the map is reduced in terms of KB, and reaches acceptable levels, with a minimum loss of detail, as in the case of ***Max density*** = 5. In other words, The ***Max density*** parameter is useful for obtaining a uniform ***PM***, and simultaneously a manageable ***.vrpmap*** file.

■ **Figura 3.199**
The higher the density, the larger the ***.vrpmap***.

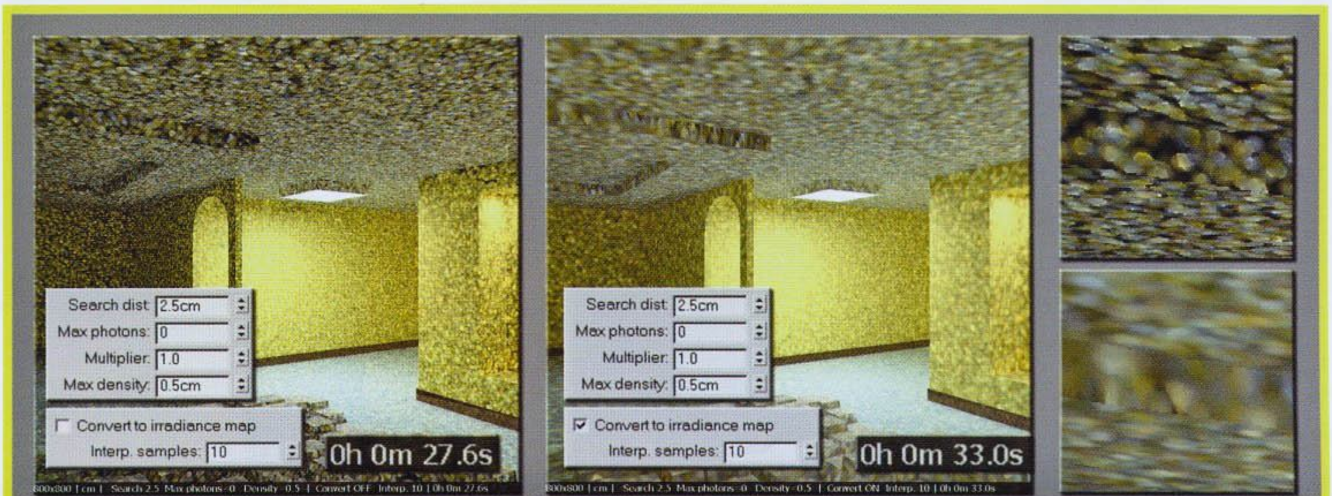




■ **Figura 3.200**
Values for ***Search dist*** from 3 to 5 times the ***Max density*** parameter are ideal for good quality renders. Different ratios generate a poor ***PM***.

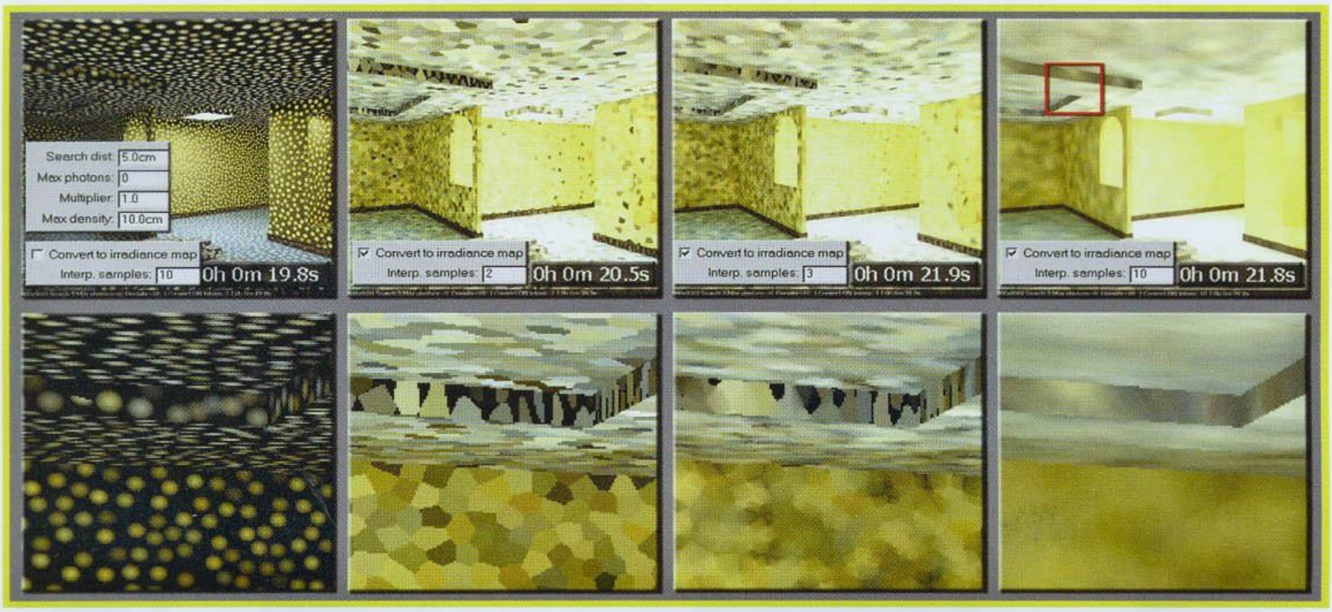
It is important to remember that changing the ***Max density*** setting requires a new ***Photon Map*** calculation. It is therefore not possible to use a previously calculated one.

Convert to irradiance map - by activating this parameter *VRay* calculates an ***Irradiance Map*** using the information in the ***Photon Map***. Take note that this is not the same ***Irradiance Map*** used for ***Primary bounces***. What actually happens is that the ***PM*** is used as an input for calculating the ***IM***. Also, the ***PM*** itself, thanks to this parameter, becomes softer. The next exercise uses the ***PM+PM*** method.



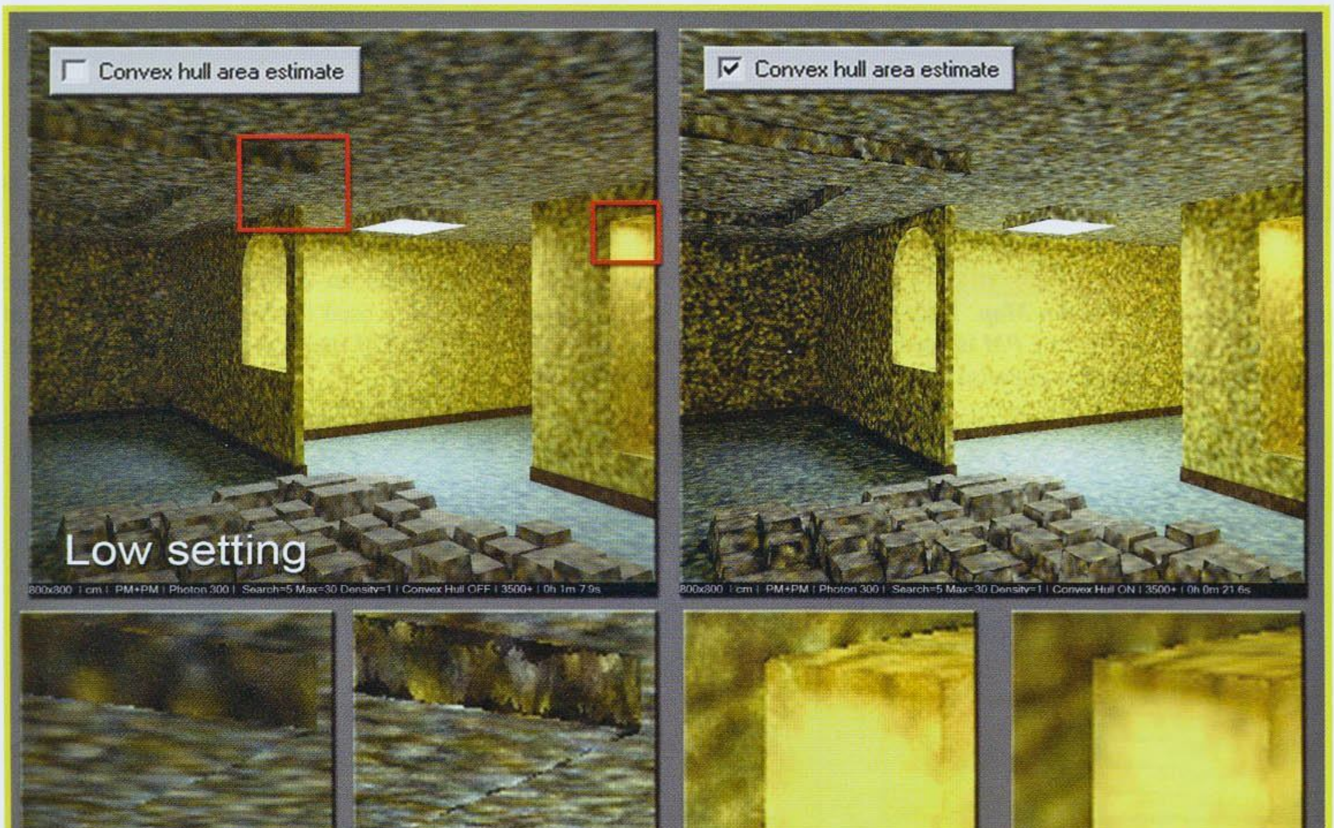
■ **Figura 3.201**
Activating the ***Convert to irradiance map*** parameter converts the ***PM*** markers into ***IM*** samples, and interpolates them. The result is a cleaner image at the expense of a bit more rendering time.

Interp. samples - the function of this parameter is exactly the same as in the ***IM***. It controls the number of samples used in interpolation, once the ***PM*** has been converted to ***IM***. High values give smoother results but also less precise. Small values produce more precise and faster results, but disturbed.

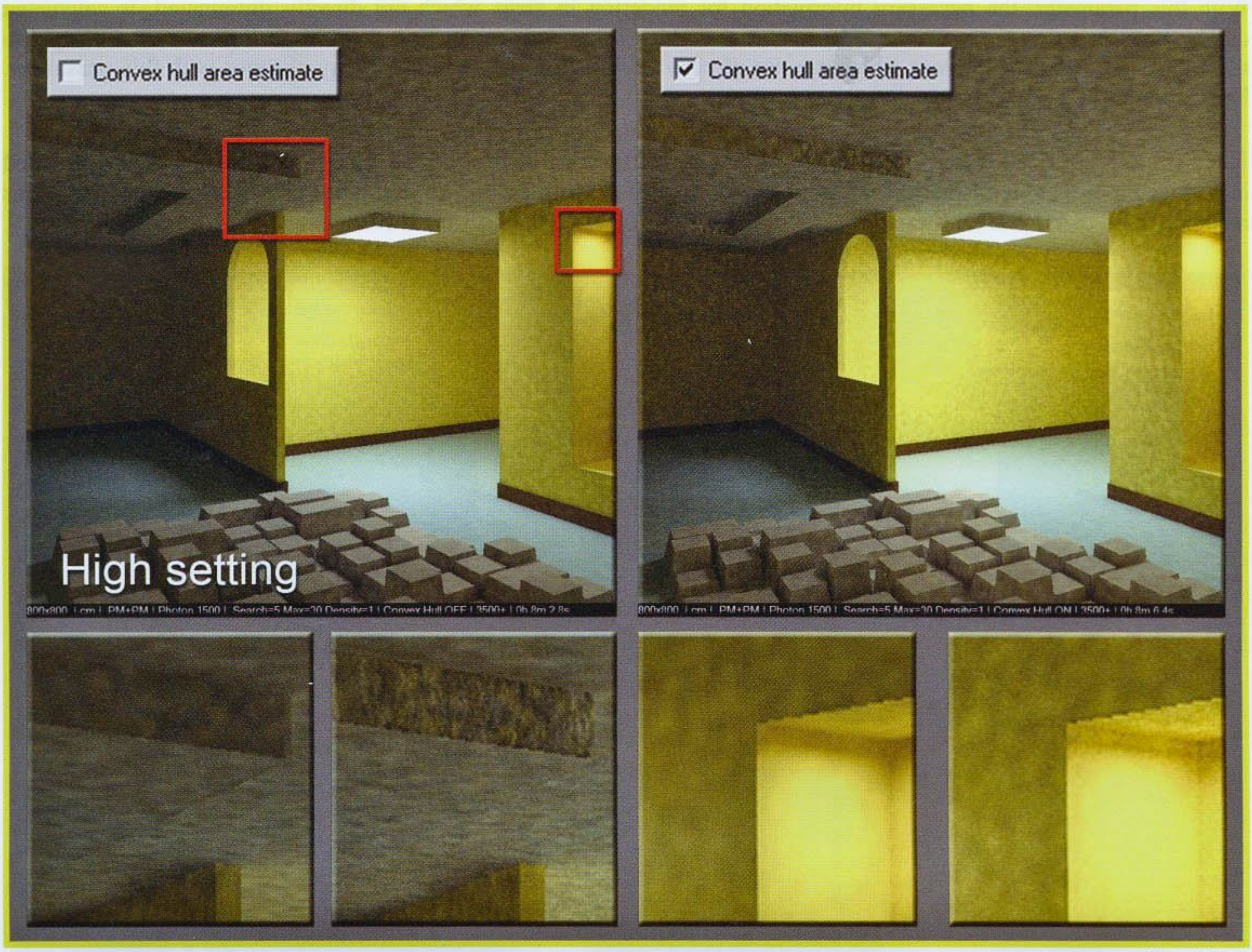


■ **Figura 3.202**
Notice the similarity between the markers and the ***IM*** samples. This is due to the conversion of the ***Photon Map*** into an ***Irradiance Map***.

Convex hull area estimate - if this parameter is deactivated, ***VRay*** uses a simplified algorithm to interpolate markers, using the distance between photons as the only reference. This algorithm is so simple that it may produce artifacts along corners. Instead by using ***Convex hull area estimate***, it is possible to avoid these problems, although it is not considered a very efficient system. Its use requires a new ***PM*** calculation.

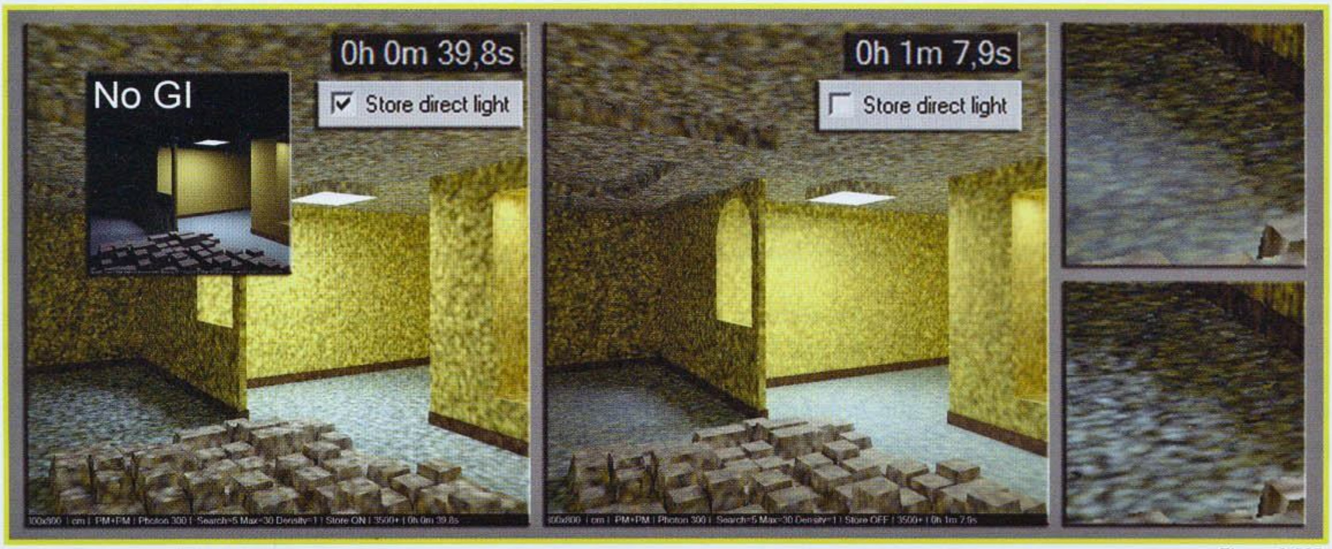


■ **Figura 3.203**
Using the ***Convex hull area estimate*** can still cause artifacts along the borders of models, especially with low ***PM*** settings.



■ Figura 3.204
With high ***Photon Map*** values, rendering improves. Artifacts are removed and edges in scenes are better lit. The ***Photon Map*** has a better lighting with a more precise GI. It is advisable to use other methods for eliminating corner artifacts.

Store direct light - when this is activated, *VRay* calculates direct light whilst elaborating the ***Photon Map***. In scenes with many lights, this operation hastens the calculation of the ***Irradiance Map*** or the ***QMC*** GI when used as ***Primary engines***. If disabled, *VRay* calculates lights via its raytracing engine. This makes their elaboration slower, especially when many lights are involved, but shadows are more precise. ***Store direct light*** is present also in ***Light Cache***.



■ Figura 3.205
In the top left render, the one with GI disabled, one can see some areas which are concerned by this parameter. Only those which are directly struck by light sources will take advantage of this option. In those zones, by using ***Store direct light***, lighting is not as precise as when this option is disabled, because the ***PM*** takes charge of illumination and is by nature more approximative. Notice though, how rendering time has been halved.

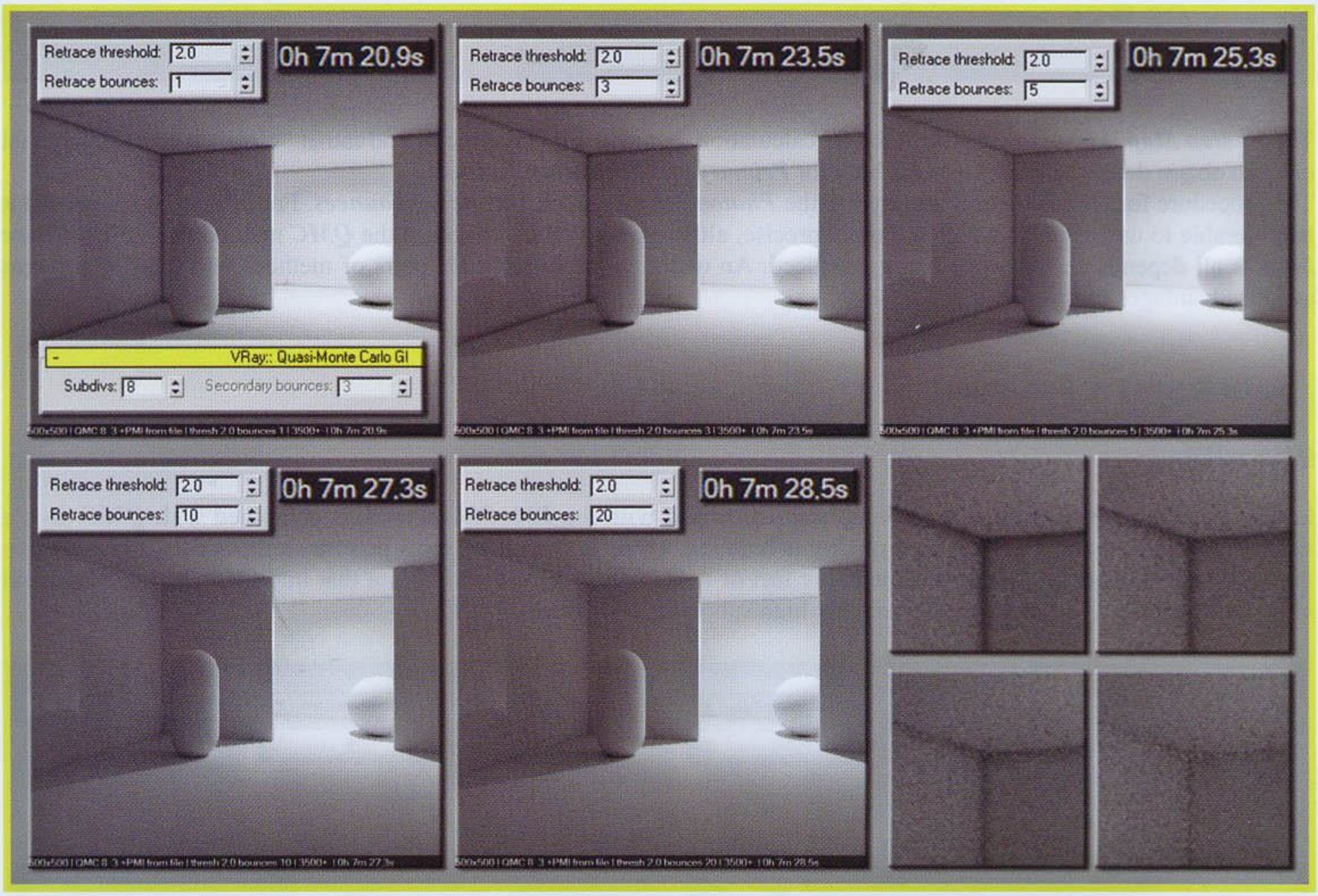
With this parameter, the necessary information for rendering in areas which are directly illuminated, is calculated during the elaboration of the ***Photon Map***, rather than during the final render. This saves rendering time. Obviously also the quality is diminished. Areas which are not directly illuminated do not undergo any changes with this parameter. Using it requires a new ***PM*** calculation.

Retrace threshold - when it is higher than 0.0, for GI calculation near model edges, ***VRay*** uses the ***QMC*** method instead of the ***PM***. One of the typical problems of the ***PM*** is the presence of dark areas and flaws near corners. This method allows one to reduce these problems, although this makes the render slower.

■ **Figura 3.206**
The ***Retrace threshold*** parameter defines the extension of the use of ***QMC*** near edges and corners.

In the test which has just been carried out, the ***QMC+PM*** was applied. With ***Retrace threshold*** = 0, only the ***PM*** is used near edges. As this value is increased, dark lines start to appear around corners. This way the freed up space is no longer for ***PM*** calculation, but is left to the ***QMC*** method. For now, these areas appear black because ***QMC*** bounces are set to 0. These bounces are set by the ***Retrace bounces*** parameter. In order to establish the number of bounces that ***VRay*** should use for GI calculation in corners, one must change this setting. The higher ***Retrace bounces*** is, the cleaner and the more flawless the image is. Using it requires ***PM*** to be recalculated.

Retrace bounces - this regulates the amount of bounces to be used by the ***QMC*** method in order to eliminate flaws along corners. If ***Retrace threshold*** is = 0.0 ***Retrace bounces*** is ignored. The meaning of the ***Retrace bounces*** parameter is identical to that of the ***Secondary bounces*** parameter in the ***quasi-Monte Carlo GI*** rollout, which instead is dedicated to Global GI.
In the next example, quite a high value for ***Retrace threshold*** has been used, in order to see the effect of ***Retrace bounces*** better. The more bounces, the better the lighting and general quality of edges is. To manage the quality of the ***QMC*** (not its bounces), one must change the ***Subdivs*** parameter which is to be found in the ***VRay: Quasi-Monte Carlo GI*** rollout. This way ***QMC*** quality for ***Primary bounces*** the same as the secondary ones used for solving corner problems.



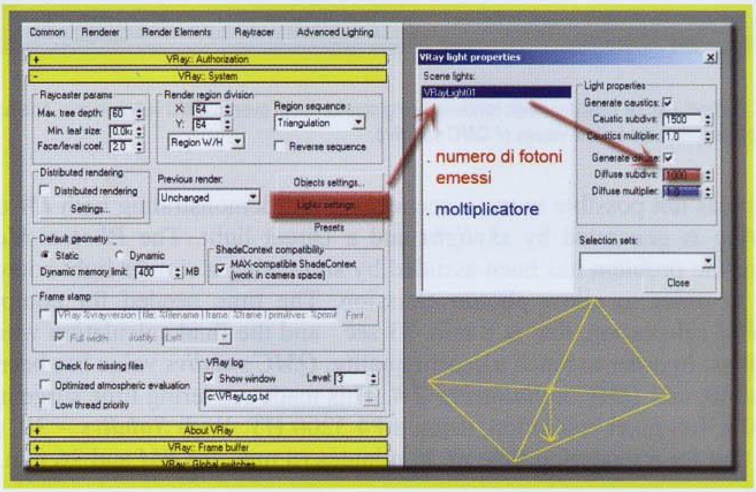
■ **Figura 3.207**
By using the unbiased method for Global Illumination along edges one can obtain better GI quality, as long as high enough values are used. This increases rendering time.

NOTES

1. Standard shaders do not generate any photons. Only with ***VRay*** shaders can one use the ***Photon Map***

2. It is preferable to use the ***Photon Map*** with light sources with inverse square decay. For this reason, if using Standard 3ds Max lighting, it is advisable to activate this in the pull-down menu ***Decay type***.

3. To control the number of photons emitted by the light source, one must intervene independently on each light, via the panel ***VRay light properties*** in the ***VRay: System*** rollout.



■ **Figura 3.208**
In the ***VRay: System*** rollout there is a button which activates a dialogue window. Here there are various options for light management. One may decide the number of photons emitted for each light source as well as their intensity.

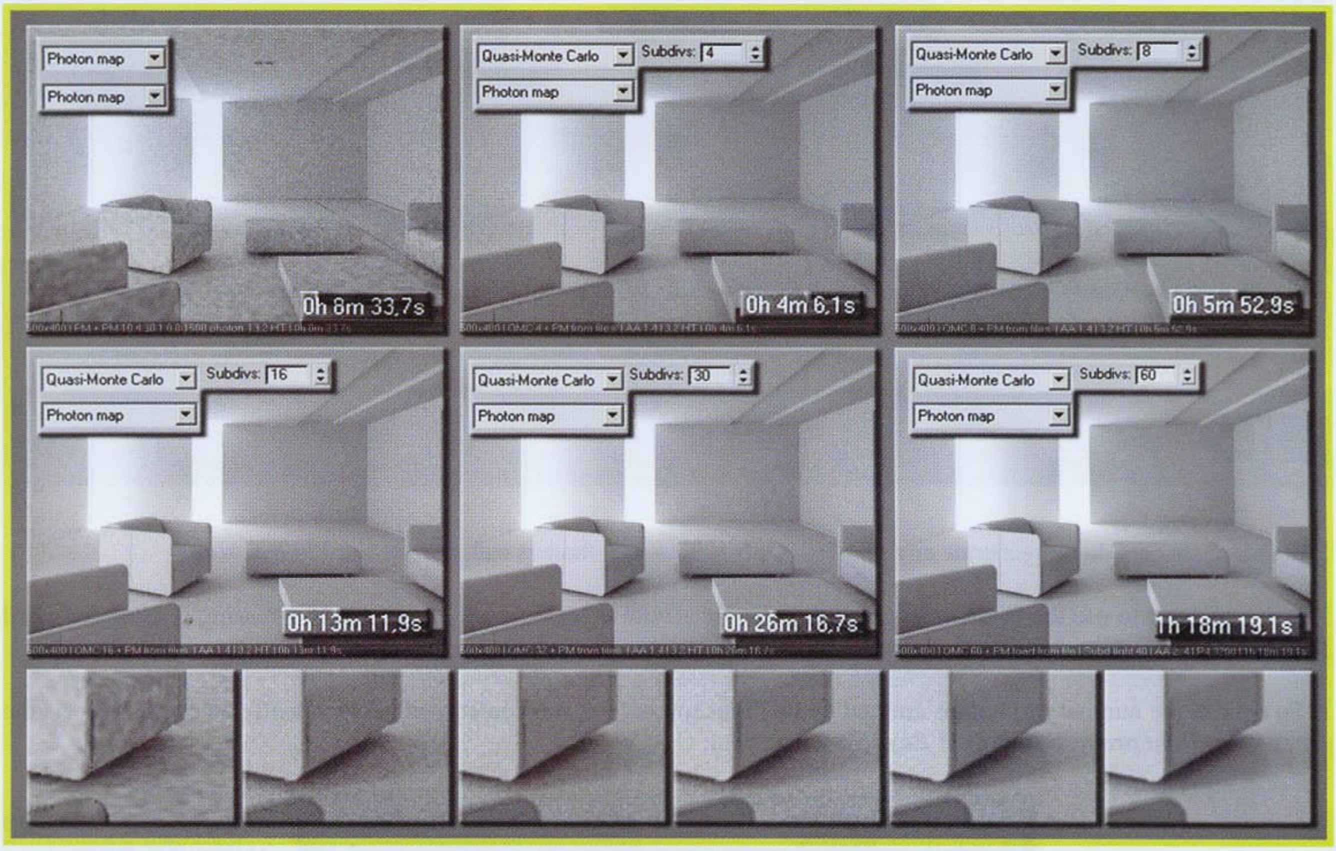
4. The ***Photon Map*** is unable to generate photons from ***skylight***, ***Environment*** or from self-illuminating objects.

The ***Photon Map*** is a very fast system for ***GI*** calculation, but also not very precise and rather "dirty". It is not likely that one can obtain decent results if using it both for ***Primary*** and ***Secondary bounces***.
The procedure for GI calculation usually has the ***Photon Map*** used for ***Secondary bounces***. For primary ones instead it is preferable to use methods which are more precise, although slower. One can use the ***QMC*** method or the ***Irradiance Map***. It all depends on the scene being produced. An overview of the possible pairs of methods and their advantages and disadvantages follows.

. INDOOR STATIC RENDERS

QMC + PM

In this case the ***Photon Map***, after having been calculated with the ***PM+PM*** method, is combined with ***QMC***. It is a system which is often used when high quality is required. ***QMC*** is one of the most precise methods of ***VRay*** and as in this case static images are being generated, the fact of being able to save the GI is not that important. In order to avoid noise in the images, it is necessary to use very high subdivision values for ***QMC***.

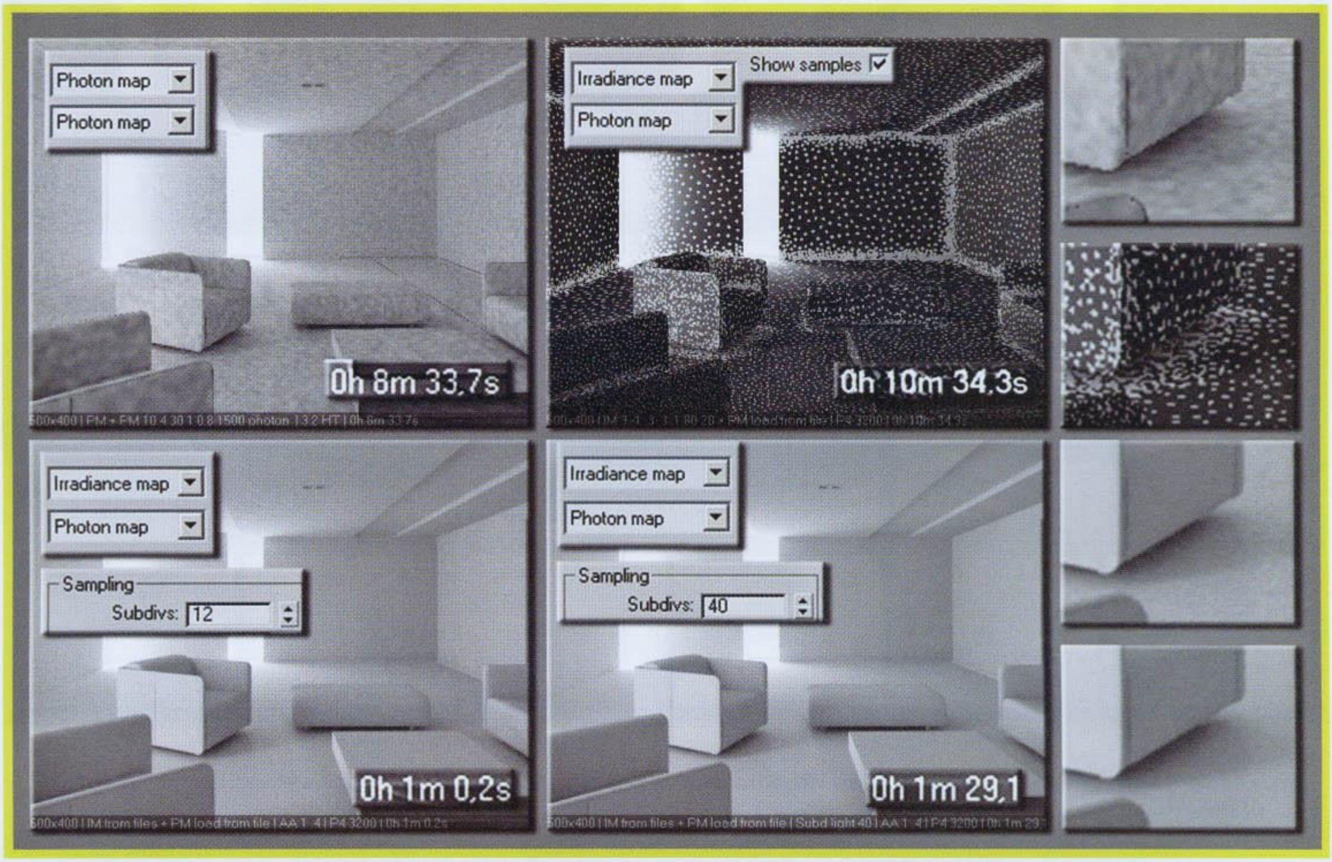


■ **Figura 3.209**
The ***PM***, combined with the ***QMC*** method, can be used for static renders, being careful not to overdo ***QMC*** subdivisions. Anti-noise filters are advisable in post-production for eliminating the problem generated by low values of ***QMC subdivs***.

In the previous example it was not possible to use a scene setup for demonstrating both ***IM*** and ***QMC*** methods. In that particular setup, the lighting is generated by ***skylight*** and a Direct light. The ***Photon Map*** unfortunately does not support ***skylight***. Therefore the problem has been avoided by simulating a similar light source. ***VRayLights*** have been placed on the openings, in order to allow photon emission. The time needed for computing the whole scene is equivalent to the sum of ***PM*** calculation, that is 8 min. 33 sec. and the final calculation which varies according to the number of ***QMC*** subdivisions. In order to reach a decent quality, ***QMC subdivs*** value has been fixed at 60, together with ***VRayLight*** subdivisions at 40, and ***QMC antialiasing*** 2 4. This made rendering time surge as far up as 1 hour 20 min. for a render of a mere 500 x 400 pixel resolution, using a P4 3200 HT. With ***Subdivs*** = 30, and no less, the result was acceptable. One can state that for a rendering time of 36 minutes (8 min. for ***PM*** and 26 min. for ***QMC*** with ***Subdivs*** = 30) the result is acceptable.

IM + PM

Thanks to the speed and approximation of the GI, the ***IM*** is one of the more used methods for indoor scenes. Its clean results and the ability to re-use the ***Irradiance Map*** favours its use for ***Primary bounces*** rather than ***QMC***. The calculation is generally made up of three phases. The first one concerns the calculation of photons with the ***PM+PM*** method. After the ***Irradiance Map*** is calculated with the ***IM+PM*** method. Once the ***IM*** has been obtained, two files are available. One for the ***PM*** and one for the ***IM***. At this point everything is ready for the final render, with very low rendering times. Also, one can re-use maps which have been previously calculated for renders larger than that of the resolution, which saves precious time.



■ Figura 3.210
The noise in this scene is essentially caused by ***VRayLights*** subdivisions. Increasing the value, it was possible to eliminate it completely.

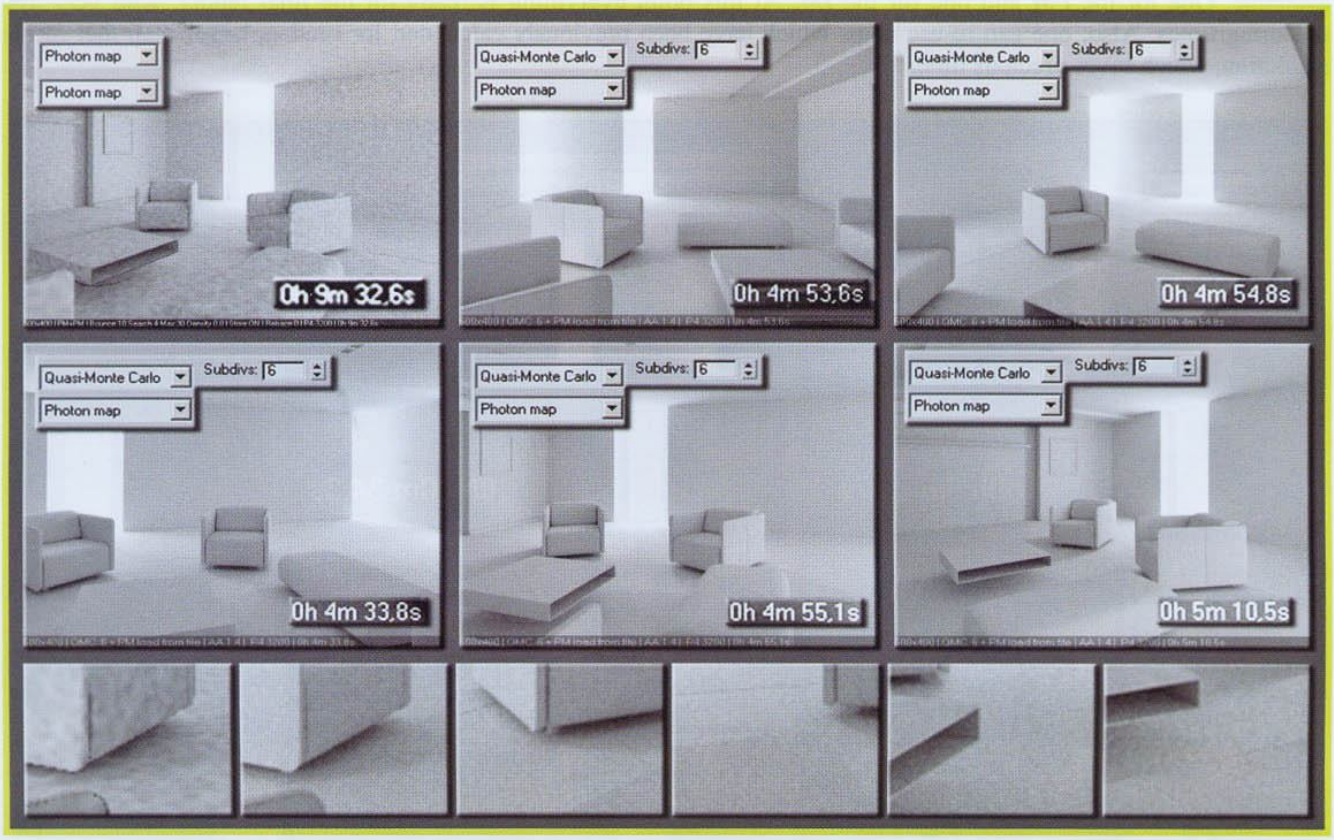
In this example the ***PM*** was used with the previous ***QMC*** method. But this time in the second phase the ***Irradiance Map*** was calculated. This last phase could actually have been calculated in the final render. But dividing the phases gives a better control over each one. Rendering time is 18 min. (8 min. for ***PM*** and 10 min. for ***IM***) with cleaner results than ***QMC+PM***.
Using the ***IM*** causes a slight general noise however. This is because of the low level of subdivisions of ***VRayLight***. This problem will be faced in detail later. For now it is sufficient to notice how changing the ***VRayLight Subdivisions*** from 12 to 40, the noise almost disappears, with only about 30 seconds extra time.

. INDOOR STATIC ANIMATIONS (Fly-through)

QMC + PM

Because static animation is by definition a series of static renders where the only variation is the point of view, it is easily understandable that what is true for static renders is also true for this type of animation. When ***VRay*** calculates the scene, it does so for the whole scene and not just the part which is seen through the 3ds Max camera. If one imagines being in a room facing a lamp which hangs in the centre, the moment ***VRay*** starts emitting photons, some of these will make their way to the wall behind the observer's back. The behaviour is similar to that of a rubber ball which bounces against everything it finds in its path leaving its mark on everything it hits. Once the ***PM*** has been calculated and loaded into the memory, one can place the camera in any part of the room: the GI will already have been calculated.

For static indoor animations with ***QMC+PM***, once the ***PM*** has been calculated all that remains to be done is change the ***QMC GI Subdivs*** value, according to the quality one wishes to obtain.
It must be said that ***QMC*** GI is not the best method for this type of render. Not being able to save any GI solutions is a problem, not so much for the quality, which is in any case high when using ***QMC***. The real problem is extremely long calculation times, especially when there is a ***Fly-through*** animated sequence. It is unlikely that for an indoor static animation one would choose ***QMC*** as a primary engine, unless one has extremely powerful hardware.



■ **Figura 3.211**
The low number of subdivisions, both of the ***QMC*** and of the ***VRayShadows***, allow average quality renders to be obtained in a few minutes.
--- DVD ANIMATION ---

The calculation time of the ***Photon Map***, around 10 min., must be divided by the 200 frames which make up the animation, which gives a result of 3 sec. per frame for a single ***PM***. A record time! To this one adds an average of 5 min. per frame during ***QMC*** and ***AA*** calculation and the result is a rendering duration which is altogether acceptable, but definitely longer than the method that we will be using briefly. In total, the animation took about 1,000 min. (5 min. x 200 frames). Roughly 17 hours.

IM + PM

In this case ***IM+PM*** is probably the best choice.

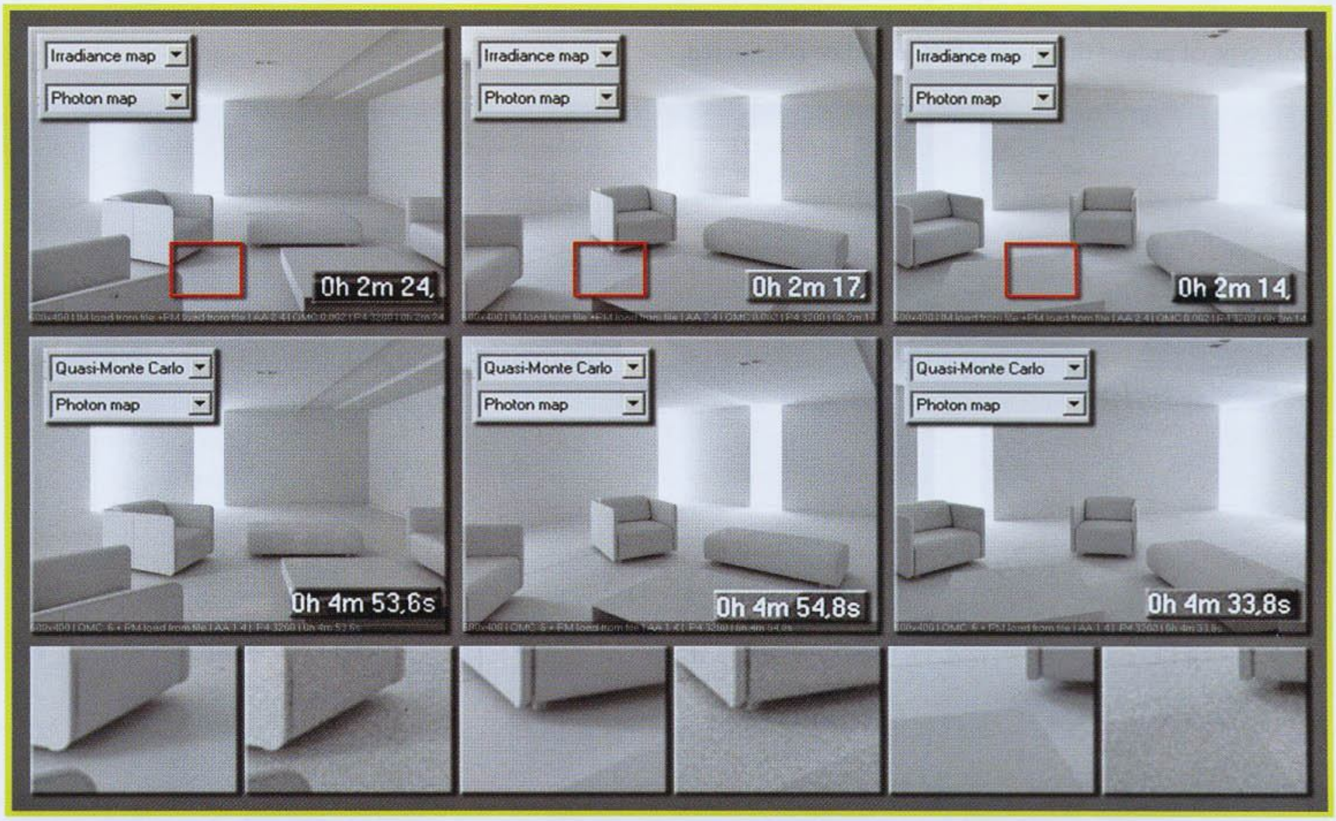
■ **Figura 3.212**
An image summarizing the parameters used for ***Photon Map*** calculation. Each of the three ***VRayLights*** produces 2,250,000 photons, and for ***VRayShadows*** 12 ***Subdivs*** were used.

The fact of being able to re-use samples in a certain frame for the following one (thanks to the ***Incremental add to current map*** parameter) is the main strong point behind the ***Irradiance Map***.

■ Figura 3.213
Because of the low animation speed, the ***IM*** has been calculated every 10 frames. This allows one to save a lot more rendering time.

The ***IM*** has been calculated for the whole sequence. The time needed for calculating the ***IM*** alone was 360 min., that is 6 hours, due to very high settings. Notice how the first frame is the one with the largest calculation time. This is due to the fact that *VRay* had not yet calculated any samples. The following frames are in fact much faster.
In order to acquire calculation timespan for each frame, one must divide 360 min. by 200 frames. This gives us 1 min. and 48 sec. roughly only for ***IM*** calculation (6 hours x 60 min. x 60 sec. = 21600 sec/200 frames = 108 sec.).
By adding 3 sec. for the ***PM***, we get a result of about 1 min. 50 sec. for the whole GI to be calculated.

At this point all that remains to be done is the final calculation, by loading both the ***PM*** and the ***IM*** into the memory.



■ Figura 3.214
The clean quality of the ***IM*** is noticeable, as is the rendering time.

Average time per frame is very low, around 2 min. 20 sec. per fame. Add this to 1 min. 50 sec. of the GI calculation using *IM+PM* and one has a total time per frame of 4 min. 10 sec. This is less than when using the *QMC* method and definitely yields a better result. Also, unlike the *IM*, the *PM* does not depend on resolution. In this case, the *IM* has been calculated with much higher values. This has allowed it to be re-used for resolutions of as much as twice as those used for its calculation, maintaining a good level of quality with much lower times compared to those given by the *QMC* method, which is definitely slower and returns a more noise.

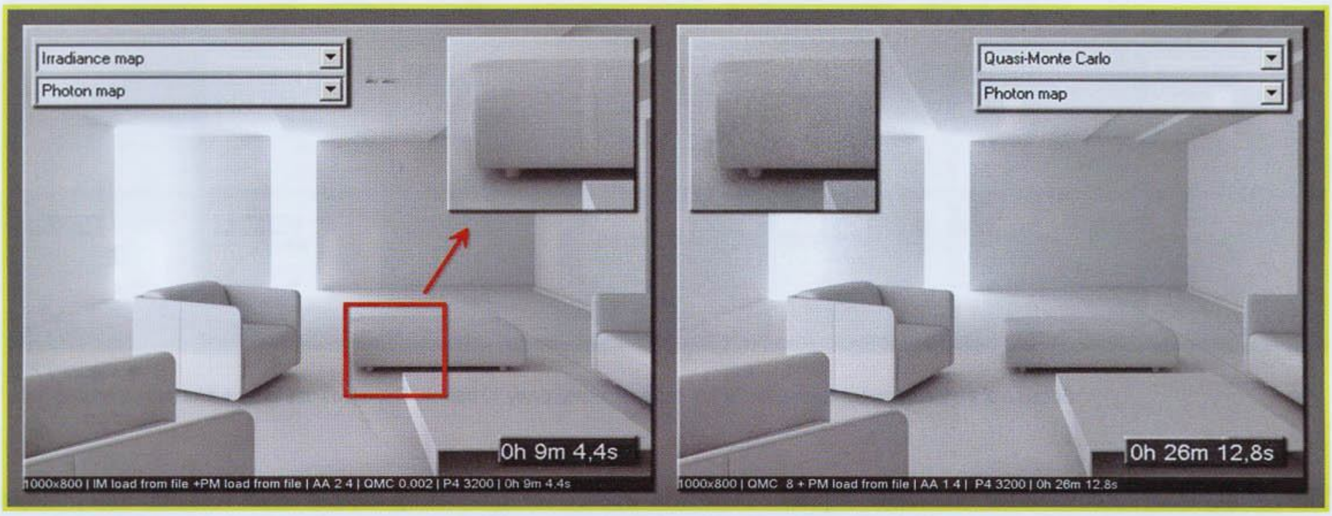


Figura 3.215
By using the *IM+PM* method, it has been possible to attain a much better result in one third of the time taken up by applying the *QMC+PM* method.

. INDOOR DYNAMIC ANIMATIONS

QMC + PM

In this type of project the use of *QMC* could be considered.

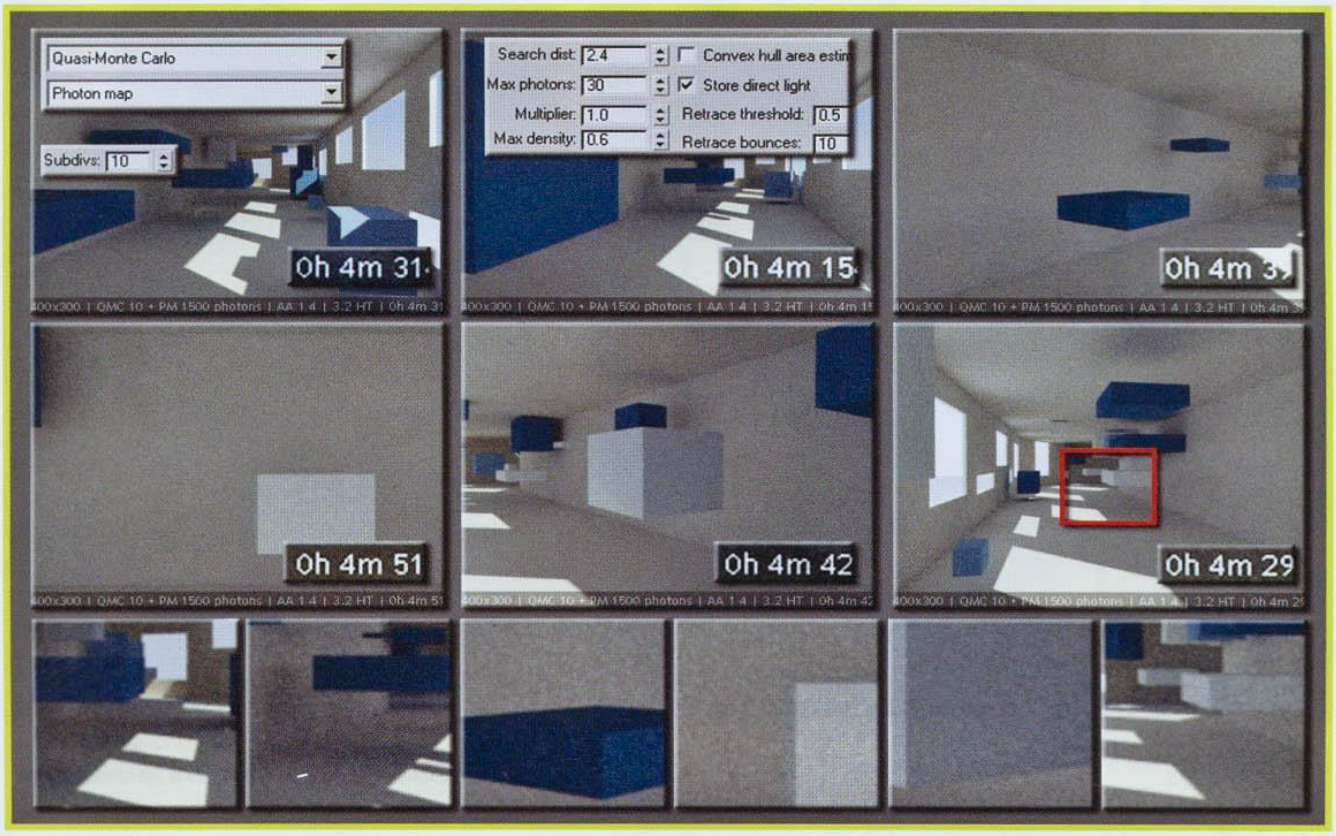
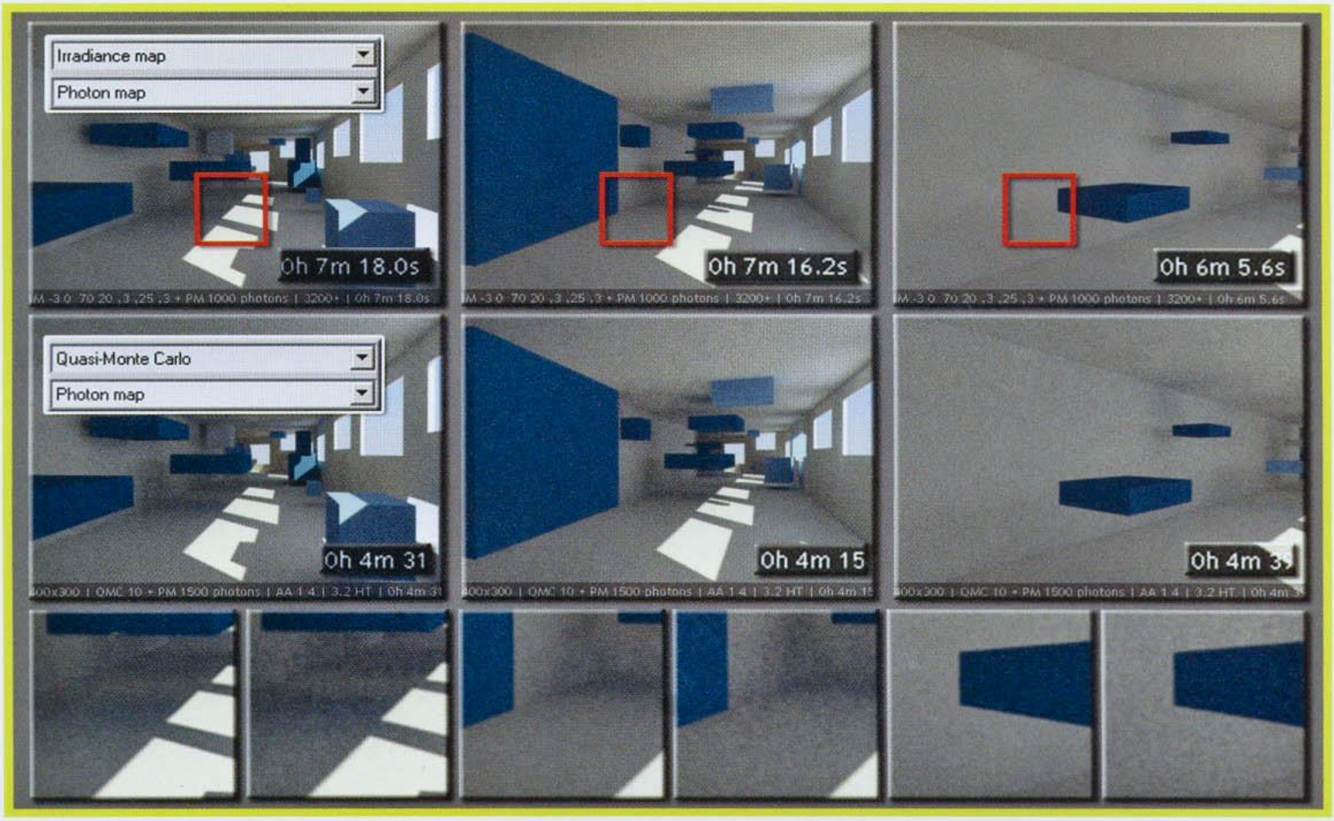


Figura 3.216
5 minutes is the time taken for calculating *PM* and *QMC*.
- - - DVD ANIMATION - - -

Saving and re-using GI maps here would not give us any benefit. The only way to render a dynamic animation is to compute Global Illumination for every single frame. For primary bounces, ***QMC*** remains in any case a method which is not advisable for indoor scenes, although very precise. Its grainy quality, which can be eliminated only with high subdivision values, makes GI calculation very slow. As far as ***PM*** is concerned, in this type of situation, secondary GI must be calculated for each frame, which raises calculation times quite a bit.
Also, still concerning the ***PM***, the fact of being able to calculate the GI in areas which are not visible to the observer is no longer an asset, but a defect, because of the waste of resources. It is of no use in a dynamic animation to have information about what is happening in an area which is not being rendered. In the following frame the situation will have changed in any case and that information would be of no use whatsoever!
From an average of about 4 min. 40 sec. per frame (280 sec.) for 400 frames, using a P4 3.2 GHz HT, one obtains a total of 112,000 seconds, in other words 31 hours!

IM + PM

The use of ***IM*** combined with ***PM*** could be a valid solution for dynamic indoor animations. Here we have two very fast methods available. Of the two, the ***IM*** is also potentially very detailed. This last aspect can be purposefully set by the user, who can decide wether to fulfill very precise using more time, or speed up calculations at the expense of quality. As in the previous example, the ***PM*** calculates markers in areas which cannot be seen by the viewer, with a considerable waste of resources.



■ **Figura 3.217**
The slight noise which one gets when using the ***Irradiance Map*** is due to the low subdivision of ***VRayLight***.
--- DVD ANIMATION ---

In this example an AMD Athlon 3200+ has been used, whereas for the ***QMC+PM*** a P4 3.2 HT. From various tests, one can see how P4 is 25% compared to the AMD 3200+. The high time needed with the ***IM+PM*** method compared to the ***QMC+PM*** must be considered, because of this, as being more or less equivalent. Therefore one can tell that, in equal times, using ***IM+PM*** also for internal animations, one can obtain much cleaner results compared to with the ***QMC+PM*** method. In this example, by using the ***IM +PM***, we have obtained an average of 7 min. per frame with an AMD Athlon XP3200+ (420 sec.), that is 168,000 sec., equivalent to 46 hours calculation!
Observing both animations (contained in the DVD), one notices the so-called flickering effect, especially near the edges in the corridor. This annoying problem is almost certainly due to a small number of photons. In this case, as in the indoor example, ***skylight*** has been replaced with a long ***VRayLight*** placed along the open wall, used for generating photons. For static renders or static animations 1,500 ***Diffuse subdivisions*** (namely 22,500,000 photons) could have been enough, in dynamic animations this number is too small. Although the parameter ***Retrace threshold*** has been activated in the ***PM***, this has not helped avoid small flaws. The only remedy is to increase the number of photons.

. STATIC RENDERS, STATIC/DYNAMIC OUTDOOR

From the previous examples we have seen how the ***Photon Map*** is extremely useful for indoor scenes. For outdoor scenes its use is not advised. ***Skylight***, a system used for simulating lighting produced by a blue sky in nature is not supported by the ***PM***. This makes it obvious why the ***Photon Map*** is useless in outdoor scenes. For this reason, test have not been carried out.

VRay: Light Cache (LC)

INTRODUCTION

Light Cache (*LC*) is one of the four methods available in *VRay* for calculating Global Illumination. It is a system invented by Chaosgroup themselves and is born from the concept of the ***Photon Map***, but without many of its limitations. The *LC*, although similar to the *PM*, for the generation of its structure generates rays which are of use for GI calculation emitting them from the point of view of the observer. In the *PM* instead rays start from the light source: basically the opposite path.
The *LC* adopts a system which is more similar to the ***Irradiance Map***. Much like the *IM*, also the ***Light Cache*** depends on the viewpoint. With the ***Photon Map***, photons are present in every part of the scene, also in areas which are not visible to the camera, similarly to what happens for radiosity. For the *LC*, samples are calculated only in areas which are visible in the moment that rendering takes place.
On the other hand, the *LC* has the characteristic of forming a cloud of points which gather information stored in a map, which is the solution used by the ***Photon Map***.

The main advantages of the ***Light Cache*** over the ***Photon Map*** are:

- Much faster scene setup. The ***Photon Map*** requires the number of photons to be decided for each single light source. With the *LC* it is enough to set the number of rays emitted by the camera. All the rest is taken care of by *VRay*.

- The ***Light Cache*** works with any light source, including ***skylight***, self-illuminating objects, lights which are not physically correct without inverse square decay, ***HDRI***, etc…

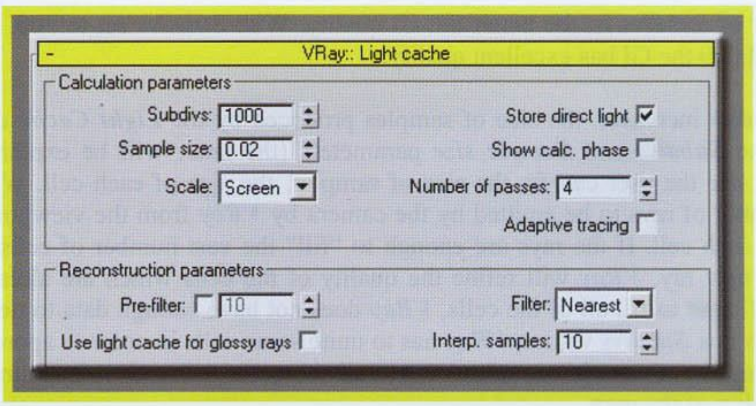
- The ***Light Cache*** produces excellent results also in tight areas, corners, edges or angles.

- The ***Light Cache*** allows one to observe GI generation while it is being carried out, just like the *IM*. It is therefore possible to see what is happening in the *VFB* without waiting for the *LC* calculation to be complete. Instead with the *PM* one must wait for *VRay* to finish the calculation and for the rendering phase to start before being able to see what is happening.

- Often the ***Light Cache*** is useful for preview visualizations, thanks to its extremely fast calculation.

The main disadvantage of the ***Light Cache***:

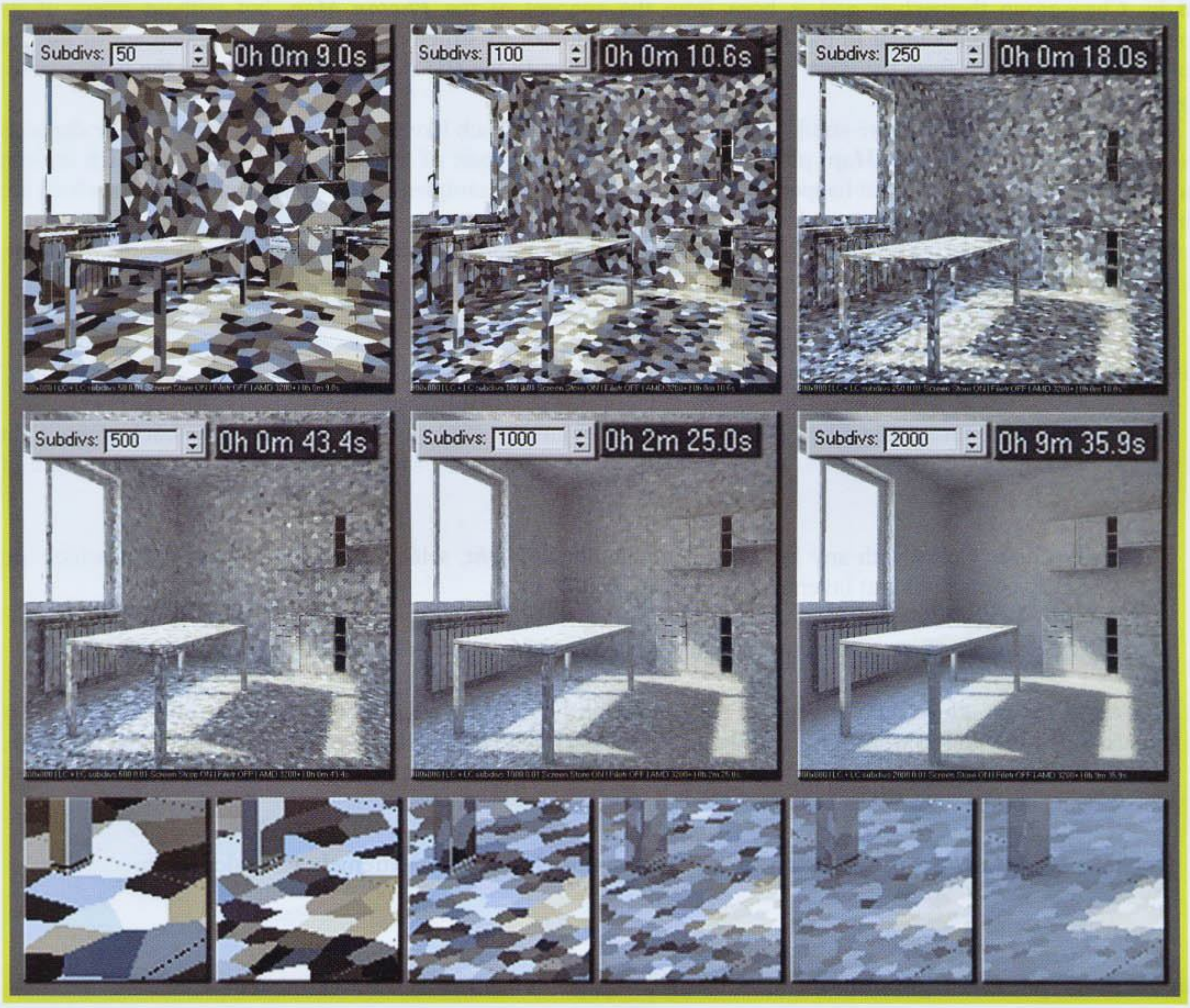
- It is a view-dependant method



■ Figure 3.218
The ***VRay: Light cache*** rollout in the Renderer panel.

Calculation parameters

Subdivs - this parameter determines the number of rays emitted by the camera. The true number of rays is represented by the square of the number shown in the ***Subdivs*** box.



■ **Figure 3.219**
As the number of subdivisions doubles, the time required for the ***Light Cache*** to be calculated quadruplicates.

1. The higher the number of ***Subdivs*** is, the better the GI quality. With ***Subdivs*** set to 500, the GI is reasonably well-defined. With ***Subdivs*** = 2,000 the GI has excellent quality.

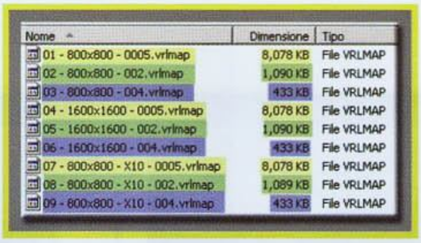
2. As the number of ***Subdivs*** increases, the size of samples produced by the ***Light Cache*** diminishes. One must take notice of the fact that the ***Subdivs*** and ***Sample size*** parameters (the latter will be explained shortly) are mutually dependant. With ***Sample size*** the user can fix the size of samples, the size of each cell. With the ***Subdivs*** parameter instead one fixes the number of rays to be emitted by the camera by ***VRay*** from the viewpoint. These rays are used to determine the "color" of each cell. If the rays are enough to "fill" the vast number of cells fixed by the ***Sample size*** parameter, then for each new ray, ***VRay*** will refine the quality of the cells which are already colored. If instead the number of rays is not sufficient to fill all of the cells, ***VRay*** does not have enough data to define the color of all of the cells. For this reason, with low ***Subdivs*** values, ***VRay*** has to unite some cells in order to approximate the GI as much as possible. But once the number of rays is enough to cover all the cells, ***VRay*** does not diminish the size of the grid further but refines the quality of the map.

Sample size - this parameter determines the size of ***Light Cache*** samples. With high values the samples are large. With low values small-size samples are generated. This way within the GI the level of detail is preserved. Moreover, the smaller the grid, the more memory is required. This is why it is best to be wary of using vales which are very small.
In order to decide the size of samples, two units of measure can be used: ***Screen*** (video unit) or ***World*** (global unit). These units influence both sample-size, as seen in the ***Subdivs*** parameter, and filters.

Screen - the image itself is used as size reference.

■ **Figure 3.220**
The size of ***LC*** samples is identical for images both with resolutions and models on different scales. It is of no importance wether the resolution is of 10x10 pixels or 5000x5000 pixels or if the image scale is in mm, cm or m. The size of the samples remains always and only defined in proportion to the image.

If ***Sample size*** value = 1, only one sample is generated of the same size as the image! This is not a particularly interesting situation. It is clear how the ***Screen*** unit uses the image itself as a unitary value, no matter what the resolution is. If ***Sample size*** is set to 0.5, the samples obtained will be the size of half the image: each sample will be 50% of the render, regardless of whether the resolution is 800x800 pixels or 1600 x 1600 pixels, as emphasized in previous tests. Value 0.1 means the samples will be 1/10 of the final image and so on… we can also observe that the physical size of the ***LC*** is the same both at different resolutions and also with models scaled up by 10 times.



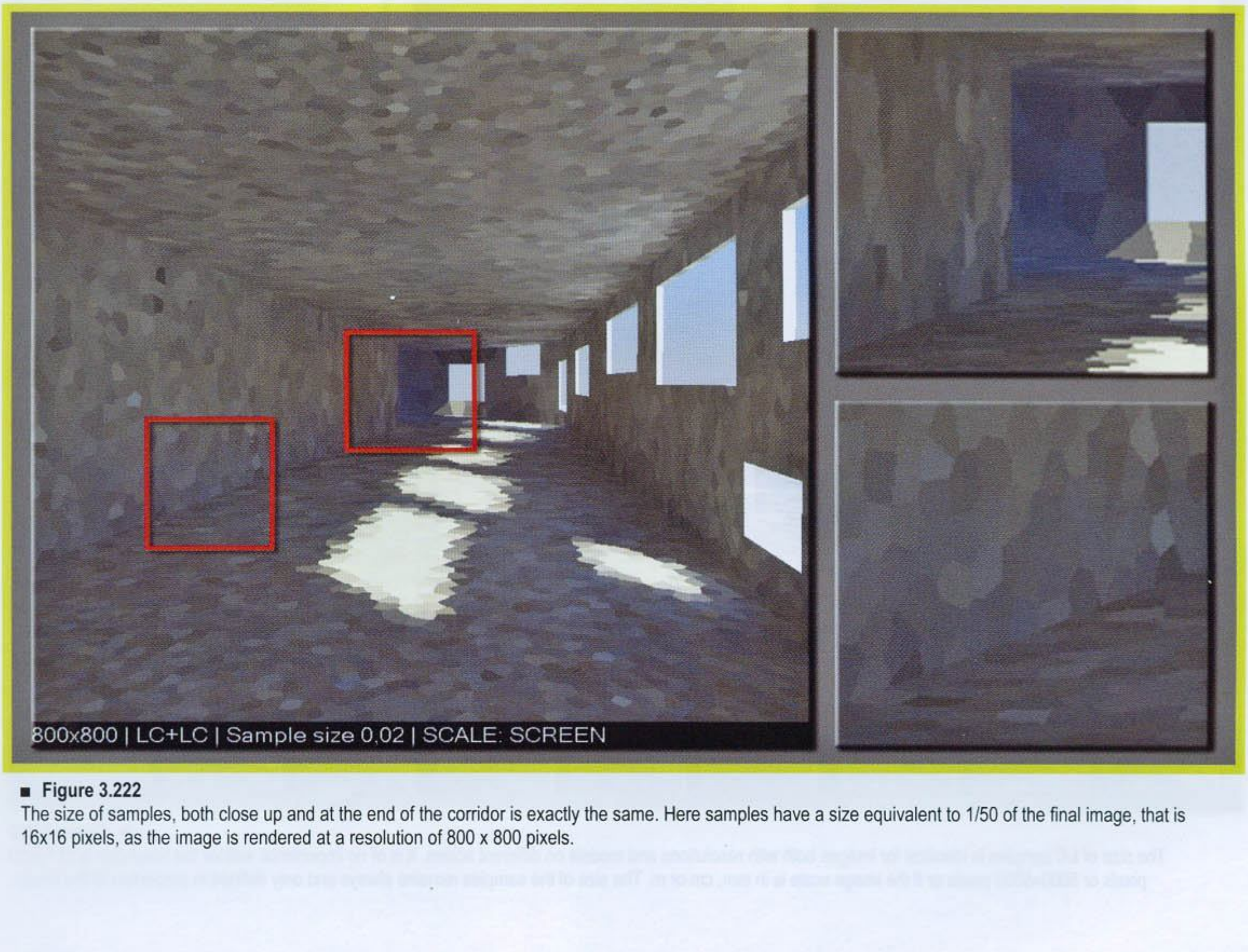
Nome	Dimensione	Tipo
01 - 800x800 - 0005.vrlmap	8,078 KB	File VRLMAP
02 - 800x800 - 002.vrlmap	1,090 KB	File VRLMAP
03 - 800x800 - 004.vrlmap	433 KB	File VRLMAP
04 - 1600x1600 - 0005.vrlmap	8,078 KB	File VRLMAP
05 - 1600x1600 - 002.vrlmap	1,090 KB	File VRLMAP
06 - 1600x1600 - 004.vrlmap	433 KB	File VRLMAP
07 - 800x800 - X10 - 0005.vrlmap	8,078 KB	File VRLMAP
08 - 800x800 - X10 - 002.vrlmap	1,089 KB	File VRLMAP
09 - 800x800 - X10 - 004.vrlmap	433 KB	File VRLMAP

■ **Figure 3.221**
As the sample-size decreases, the size of the ***LC*** gets larger, the more sample-size decreases. This is because the ***LC*** has to store a greater number of samples per surface unit. The smaller samples are, the more there are.

In the light of what has been explained so far, two observations can be made:

1. The size of samples does not differ, regardless of wether they are at the end of a corridor or close to the viewer. This explains why ***Screen*** measure is not advisable for ***Fly-through*** animations. In fact in a static animation, the closer one is to an object, for example the wall at the end of a corridor, the more samples would appear on the surface at each change of frame. This happens because the wall adapts to the new amount of space it takes up in the image. If, when the wall is far away from the camera, it is made up of one sample only, as the viewer draws closer the samples which make up the wall grow more numerous and this may create flaws including the famous flickering effect.

2. For static images it is preferable to use the ***Screen*** unit, because for models which are far away from the scene and thus not extremely visible, the samples calculated for those objects are very few. This avoids wasting time in calculating samples which are far away, extremely small and not of major interest.



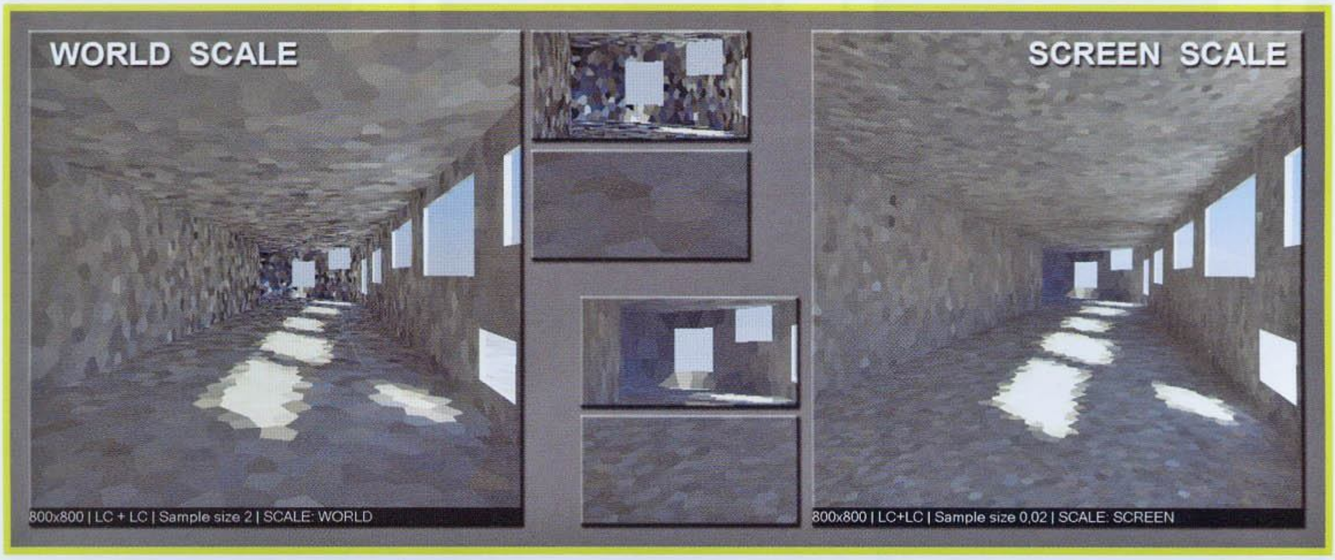
■ **Figure 3.222**
The size of samples, both close up and at the end of the corridor is exactly the same. Here samples have a size equivalent to 1/50 of the final image, that is 16x16 pixels, as the image is rendered at a resolution of 800 x 800 pixels.

World - with this measuring unit file size is used. Once the file is set in cm, the size of samples is also expressed in cm, etc…

■ **Figure 3.223**
Using ***World*** as the measuring unit, the size of samples is tied to the geometric size of the image.

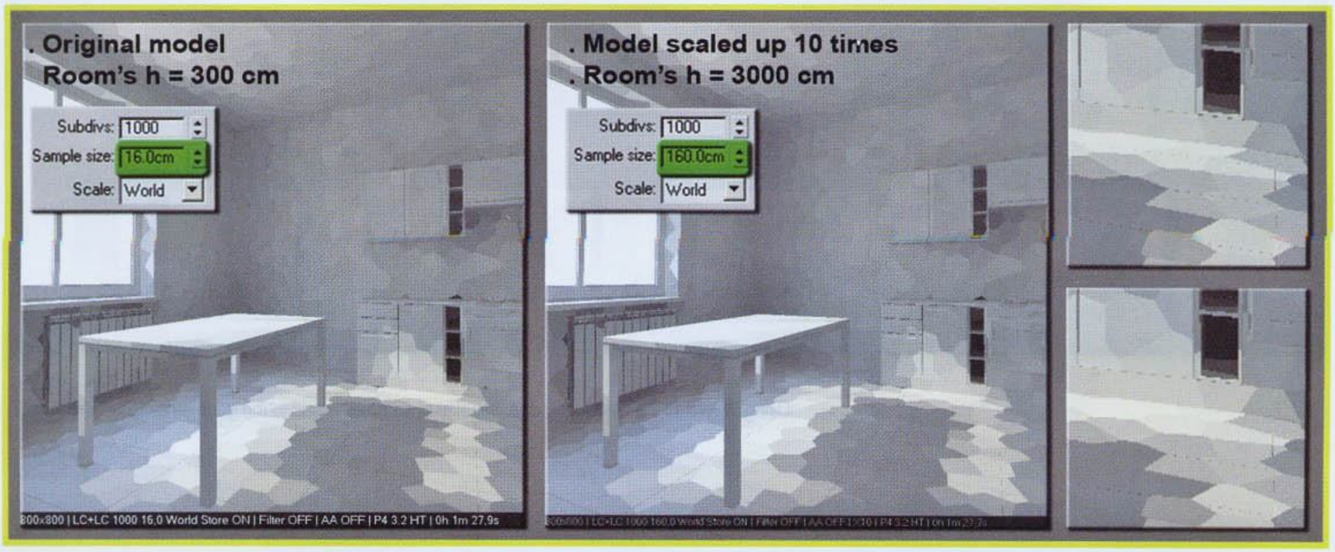
Compared to the ***Screen*** unit there are two main differences:

1. With the ***World*** measurement samples have an actual size. Wherever they are, near or far from the camera, they have a physically measurable size. This means that when rendering, the samples closer to the camera are larger than the ones further away, although they mutually share the same size. While the ***LC*** is being calculated in a ***Fly-through*** animation, when moving closer to a wall, samples do not change size but remain the same. It follows that for animations, the ***World*** unit is preferable to the ***Screen*** unit.



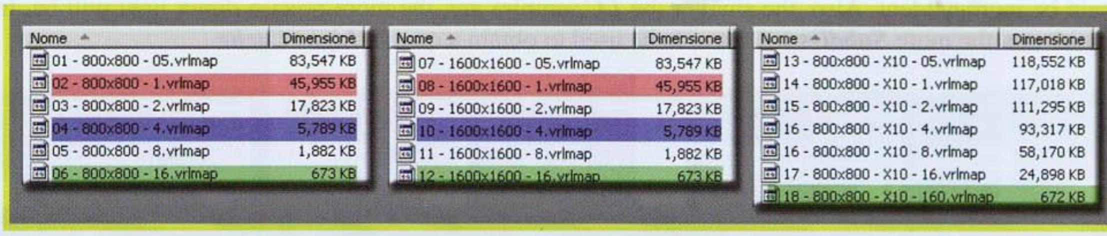
■ **Figure 3.224**
Using ***World***, the samples closer to the camera seem larger, and the ones further away seem smaller, although in truth they are the same size. In static renders, high detail in far away areas is superfluous, as they are scarcely visible.

2. With the ***World*** unit, the size of samples depends on the physical scale of the model. If for any reason the scene were to be re-sized, enlarged or made smaller, in order to obtain the same sample-size before transformation, also the ***World*** unit would have to be changed.
If one had to calculate the ***LC*** for a room 300 cm high, one would have to use a ***World*** value of 3-4cm. If the room were scaled up 10 times, one would have to calculate the ***LC*** for a model 3000 cm high! If one had to use the same ***World*** value (= 4cm), the ***LC*** would have samples so small that they would be almost invisible. So the appropriate thing to do is change ***World*** to = 40cm.



■ **Figure 3.225**
Two identical GI solutions with models of different sizes. For this reason, the ***Sample size*** parameter has been changed to suit the new model size.

By analyzing the file produced by the ***Light Cache*** one can draw some interesting conclusions:



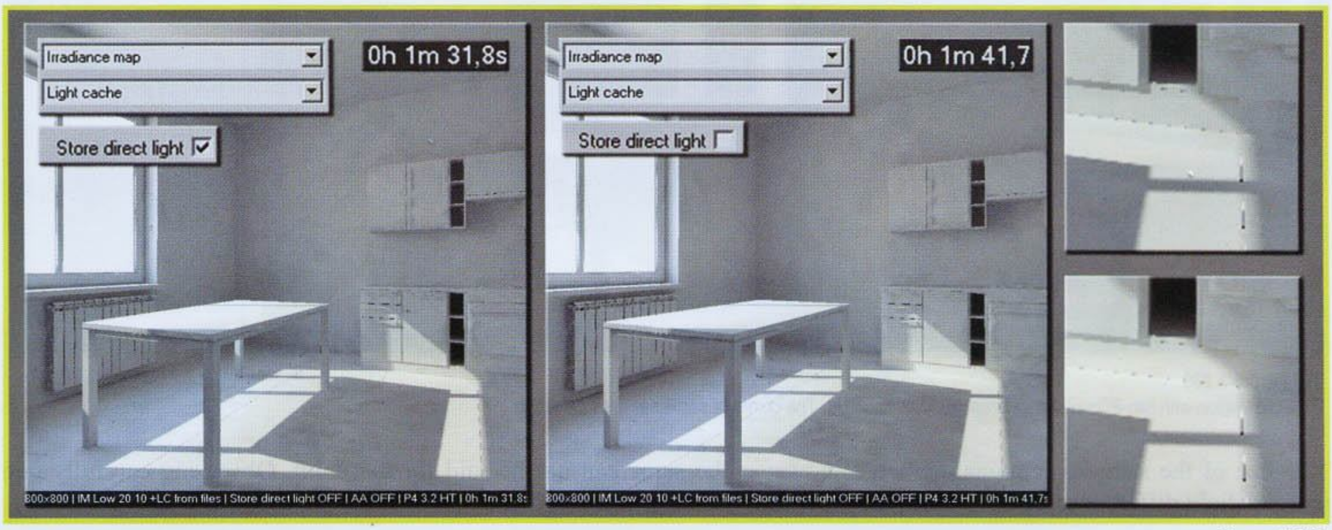
■ **Figure 3.226**
Size in KB of the ***LC*** from the previous example.

The first is that the smaller samples are, the greater the file for the ***LC*** map, regardless of output resolution. The second is that as the size of the model doubles (in our case it is ten times), in order to obtain the same ***LC***, the size of the samples must be doubled as well. For an original model with ***Sample size*** = 16, enlarging the scene x10 means using ***Sample size*** = 160. In both cases, the ***LC*** weighs 672 KB.

Store direct light - activating this parameter makes the ***LC*** calculate direct light as well as the GI. ***Store direct light*** is useful in a scene with many lights. This way in fact, ***Light Cache*** takes care of direct light calculations, making the rendering process faster. If the aim is quality renders, ***Store direct light*** must be disabled. Doing so means that the direct light is calculated during the final render, taking up more time.

■ **Figure 3.227**
In this case, with a simple Direct Light, the time taken has gone down by 30 sec. on a total of 2 min. An excellent percentage.

Observe the use of this option together with ***Irradiance Map*** used for ***Primary bounces***.



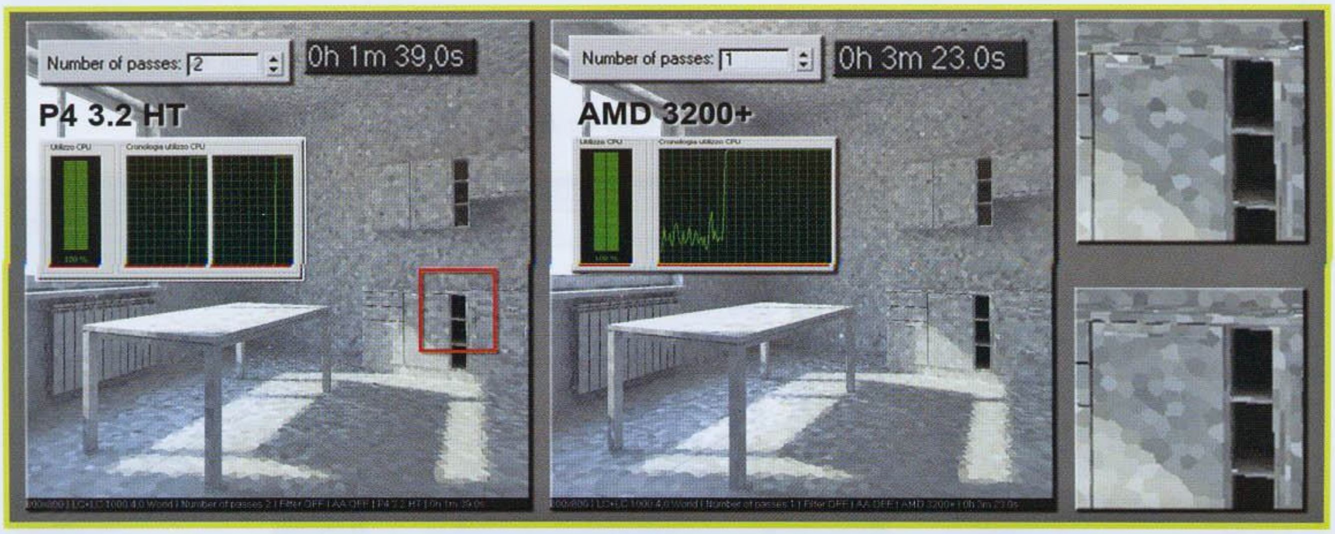
■ **Figure 3.228**
The effects of the ***Store direct light*** parameter are visible only during ***LC*** calculation. Rendering time has gone down by 10 seconds.

The ***Store direct light*** reduces rendering time. In the case we are examining, there are only two light sources, so time reduction is not so consistent. In scenes with many lights the benefit drawn by using the ***LC*** for calculating direct light would be more evident. Also, the smaller the ***LC*** samples are, the more precise direct light will be. Instead the smaller samples are, the more ***Subdivs*** will have to be used to obtain a quality ***Light Cache***.

■ **Figure 3.229**
Example of variations in the quality of shadows on the basis of an increase in ***Sample size*** in the ***LC*** when using ***Store direct light***.

Show calc phase - activating this parameter, the user can watch ***Light Cache*** calculation inside the ***VRay Frame buffer***. This option has the purpose of giving the user visual feedback on what is happening. This parameter does not influence ***Light Cache*** quality in any way.

Number of passes - the ***Light Cache*** is internally calculated by ***VRay*** in different phases, invisible to the user, which are merged at the end. Each phase is calculated aside from the others. By using this method, one can generate a ***Light Cache*** with the same appearance on different machines with a different number of CPUs.
In the case of a dual CPU single core, 2 ***threads*** are available. Each one calculates a single phase, which is then merged with the phase calculated by the other CPU, and therefore generating the final ***LC***. In this case one should set the ***Number of phases*** (passes) to 2, just like the number of ***threads***. If one is using only one single core CPU, thus having only one ***thread*** available, it might seem correct to set the ***Number of passes*** to 1. If we look carefully though, the ***LCs*** produced by the two PCs have slight differences.

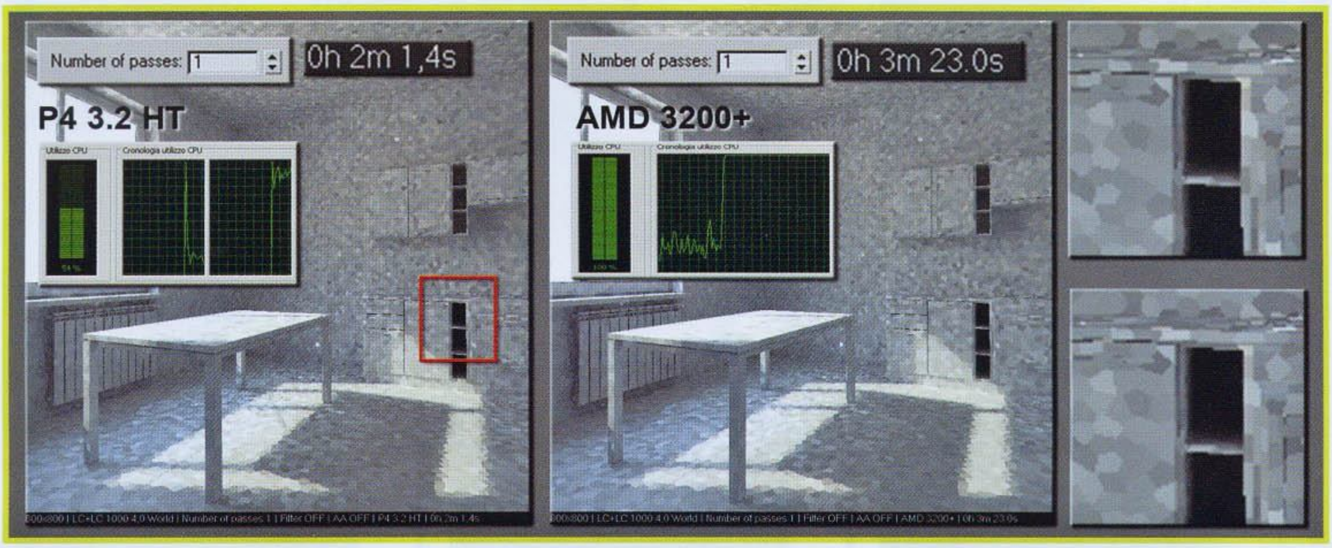


■ **Figure 3.230**
LC calculation with two PCs with a different number of CPUs. The colors of some samples different.

The size of the samples remains unaltered; however the same can not be said for the color. Observing carefully one notices slight differences. In both cases the PCs have been exploited 100%.

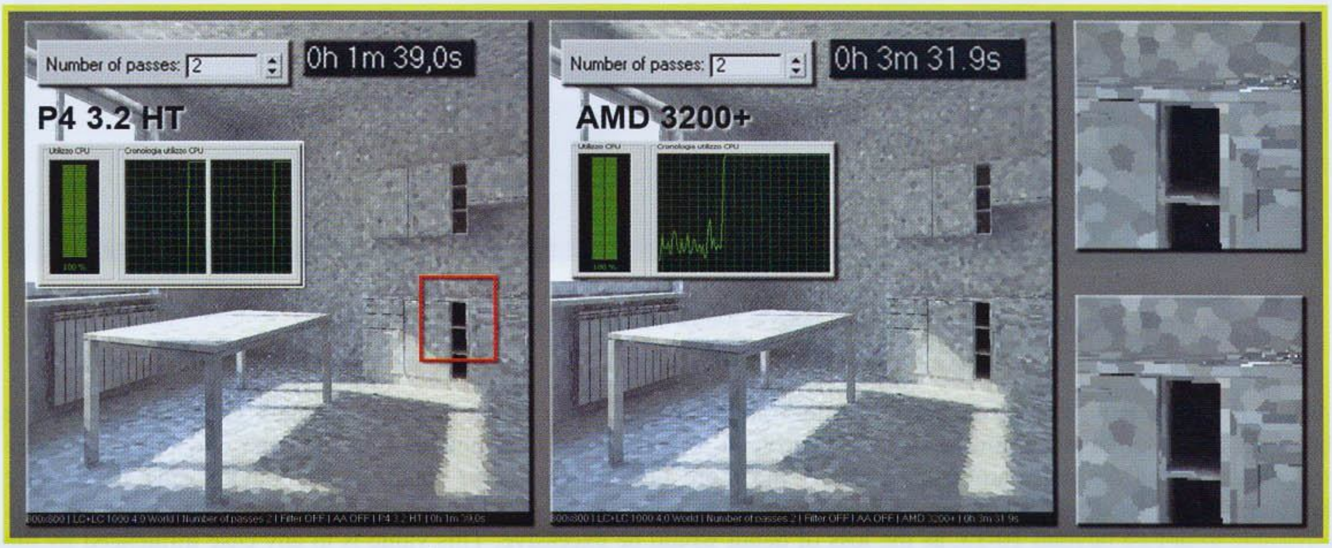
This phenomenon takes place because the calculation of two different ***passes*** happens in a slightly different way. The possible solutions for having the same ***LC*** with the availability of two different computers, are two:

1. Set the ***Number of passes*** = 1 for the PC with 2 CPUs. However this would mean that the second CPU is inactive, as shown by the task manager.



■ **Figure 3.231**
With two different hardware conFiguretions one can obtain the same ***LC***. In this case though, we have opted for the deactivation of one of the ***threads*** for the Pentium 4 HT. This creates a higher rendering time.

2. Change the ***Number of passes*** of the single CPU computer, and set it to 2 ***passes***. This is the best choice, because both machines are exploited at their full potential and produce the same ***LC***.



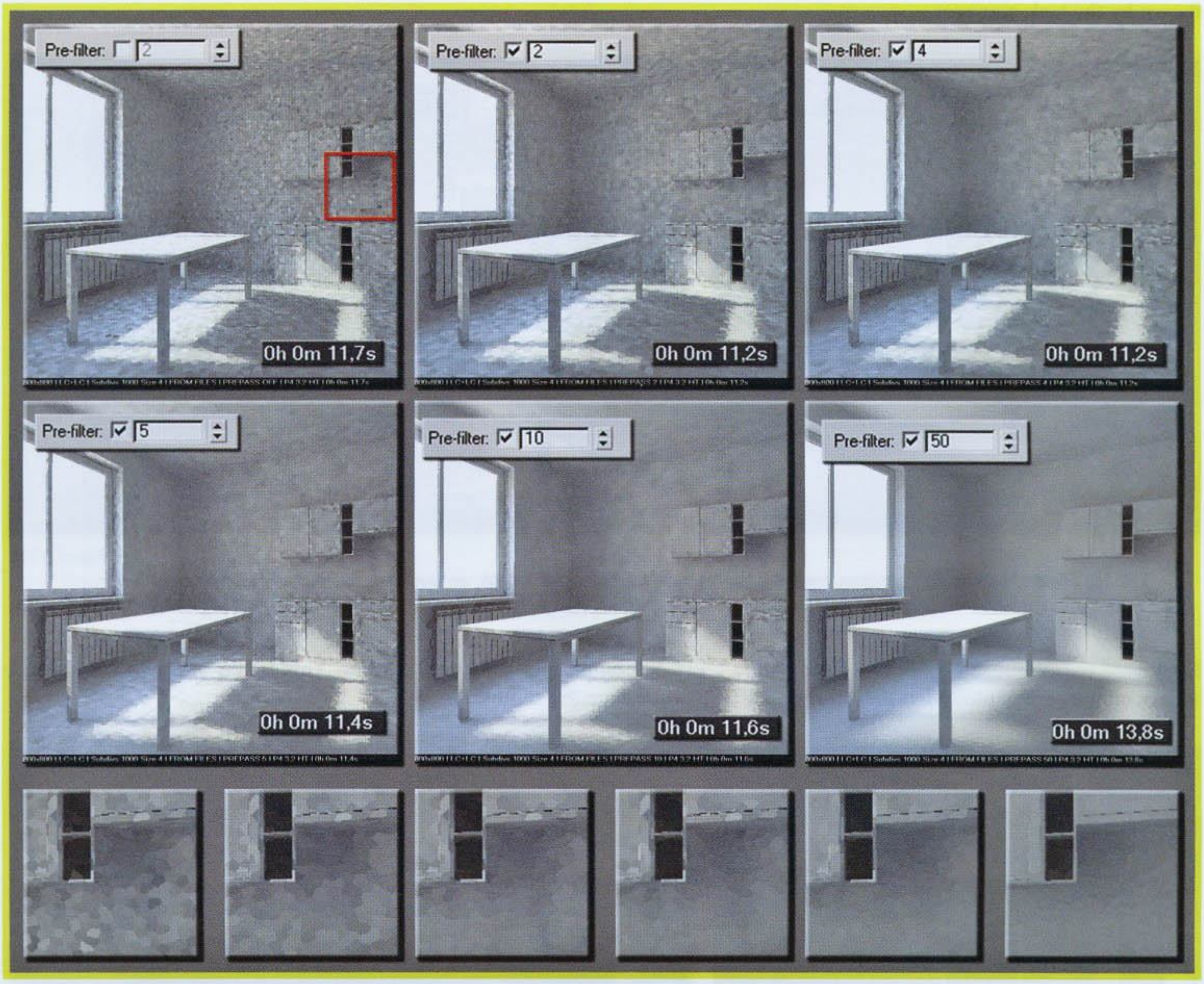
■ **Figure 3.232**
By using the number of PC passes with more ***threads***, one can obtain the same ***LC*** from machines with a different number of CPUs.

In general, when the ***Light Cache*** is calculated with less ***passes***, the result is less disturbed and cleaner. On the other hand, with few passes, the efficiency of multi-processor computers or networks is reduced.

Reconstruction parameters

Pre-filter - when this parameter is activated, samples in the ***Light Cache*** are filtered before final rendering. Take care not to confuse this filter with the ***filter*** options (see following pages) which instead filter the ***Light Cache*** during the actual rendering process.

Pre-filter examines samples one at a time and modifies their properties analyzing nearby samples, therefore calculating the average. The more samples are used for the ***Pre-filter***, the more uniform and flawless the ***Light Cache*** is. These calculations can be carried out also once the ***Light Cache*** has been calculated. So it is possible to calculate it, save it, load it and then experiment with different values.



■ **Figure 3.233**
The ***Pre-filter*** modifies color and intensity of each ***sample***, applying an average with nearby ones, but maintaining its perimeter.

As the ***Pre-filter*** parameter is changed, also the samples of the ***Light Cache*** undergo changes. Color, saturation and brightness of samples are changed so that they blend in better with their neighboring samples. Obviously the more samples are used for calculating the average, the more uniform the ***Light Cache*** is, although losing detail. The minimum value for ***Pre-filter*** is two. In fact in order to calculate an average at least two samples are required. It is advisable to not go beyond 10-15, as the loss of detail would be excessive. This value is in any case influenced by the size of samples. The smaller are the higher ***Pre-filter*** values can be applied, because the level of detail is preserved by the high number of small samples. The larger the samples are, the lower ***Pre-filter*** should be set.

Filter - these filters are applied during the final rendering process. They determine the way samples in the ***Light Cache*** are to be interpolated with nearby samples.

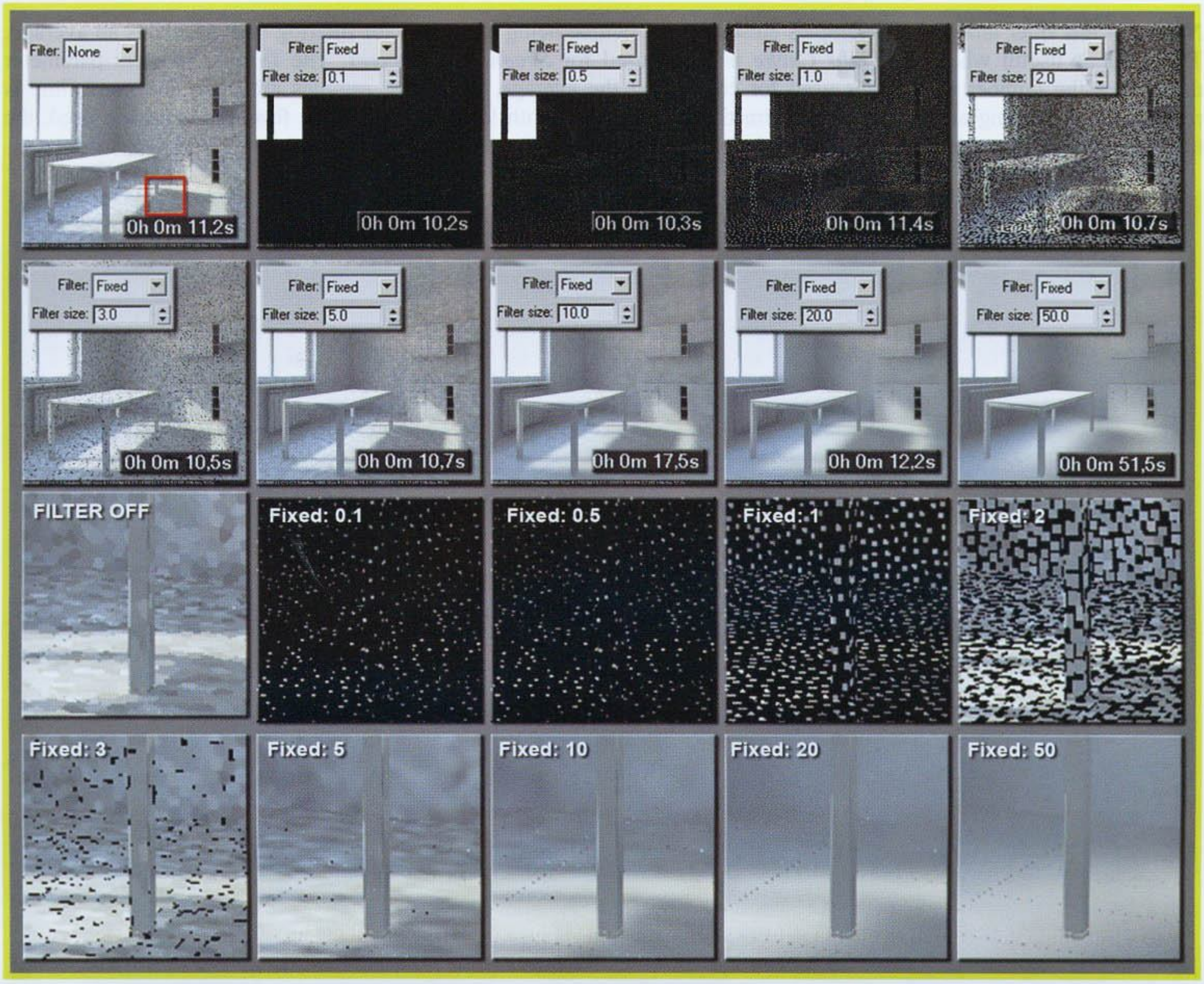
None - no filter is applied. Samples are not filtered and their value is taken accurately. This is the fastest method as *VRay* does not apply additional calculations, other than taking the samples and rendering them, using the ***Light Cache*** data. However, this method may cause artifacts near edges or noise near geometrical objects. It may therefore be useful to use ***Pre-filter***. It is advisable to keep filters disabled only when the ***LC*** is being used for trial renders.

Nearest - this filter analyzes the samples closest to the sample being examined for filtering and calculates an average. This filter is useful when the ***Light Cache*** is being used for ***Secondary bounces***. The ***Interp. Samples*** parameter determines the number of samples used for calculating the average. The more samples are used, the more time is required for rendering and the more uniform the ***LC*** becomes, with loss of details. The fewer samples are used, the faster the rendering process is with a higher level of detail but also more artifacts.

■ **Figure 3.234**
Variation in ***Light Cache*** quality whilst changing the ***Nearest*** filter values. The samples are blended with the ones nearest to them.

The ***Nearest*** filter, as well as being calculated in the final rendering phase, blends the samples and makes the ***LC*** more uniform. The effects of filtering are visible with values greater than 1. It is in fact impossible to average with a single sample.
Optimal value goes from 5 to 10 samples. Beyond these values problems may arise such as loss of detail. This problem can be observed near the table leg in the image, which seems not to touch the ground because of the high level of blending in the shadow on the floor.

Fixed - this filter analyzes and calculates the average of all the samples near the one being examined, on the basis of their distance from it. This can be useful when using the ***LC*** for ***Primary bounces***. The size of the filter is managed by the parameter ***Filter size***. High values make the GI more uniform, eliminating artifacts at the expense of lower detail. In most cases, ***Filter size*** should be 2 or 4 times the ***Sample size***. The ***Filter size*** parameter can be expressed in Screen or ***World*** units, according to the settings on the ***Scale*** parameter.



■ **Figure 3.235**
Variation in ***Light Cache*** quality due to changes in the ***Fixed*** filter.

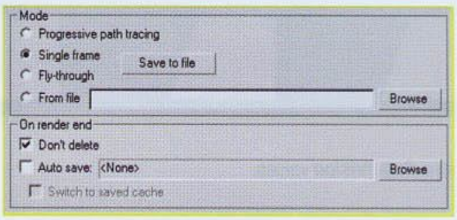
As the size of the filter is increased, the ***Light Cache*** undergoes a blending effect. The best conciliation between uniformity and detail loss is a value of around 10 cm, that is more than twice the vale in the ***Sample size*** parameter in this example, which is 4cm. With higher values we obtain a more uniform ***LC***, but problems start to appear, due to the use of an overly long radius, which causes problems such as "Flying objects" and strange illumination effects. With a high ***Fixed*** value, the GI is distorted by samples which are too far away and alter its value excessively.

Use light cache for glossy rays - with this parameter activated, the ***Light Cache*** is used for calculating for the ***glossy*** effect of certain shaders. This can speed up rendering greatly in the presence of shaders with blurred reflections or refractions.
When ***Light Cache*** is used alongside ***QMC*** or ***Irradiance Map***, one can use the option ***Use light cache for glossy rays.*** The increase in speed is considerable and likewise the quality benefits, as long as ***LC*** is high quality. Otherwise, the ***glossy*** rays could bear flaws.
In the following example a scene is rendered with the ***LC+LC*** method. After, the ***IM*** is used as Primary bounce engine and the ***LC*** for ***Secondary bounces***. Rendering time, thanks to the use of ***LC*** for ***glossy*** effects, is more than halved!

■ **Figure 3.236**
Thanks to the use of ***LC*** also ***glossy*** shaders, which are usually slow to calculate and render, can be quickened and still have excellent quality.
--- DVD ---

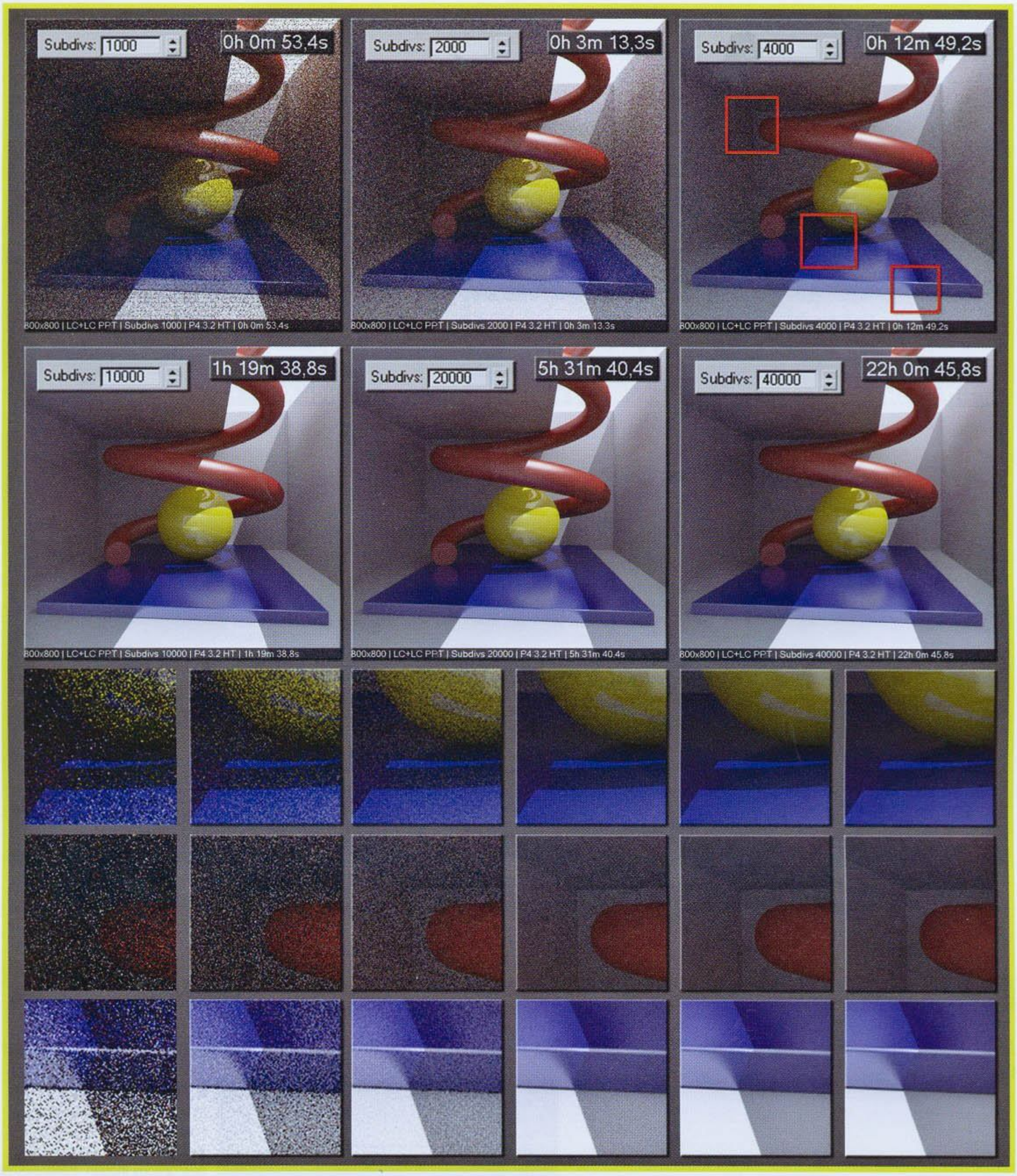
Mode

Just like the ***Irradiance Map*** and the ***Photon Map***, also the ***Light Cache*** produces files storing the information about the GI.



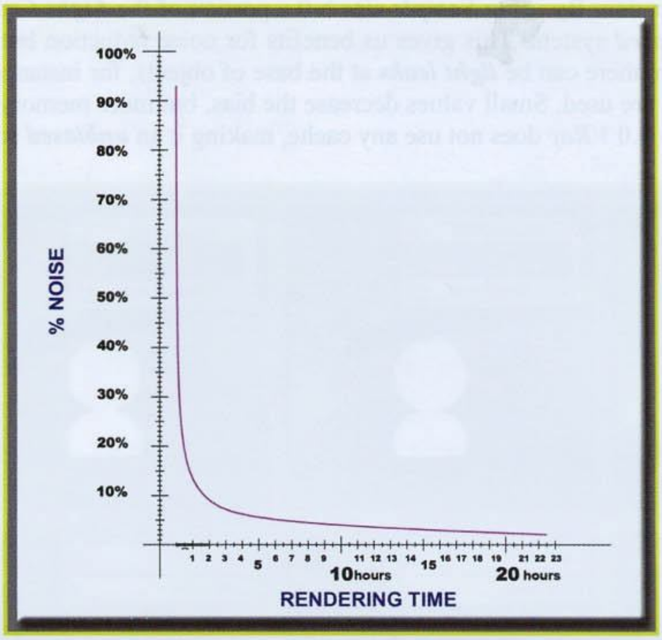
■ **Figure 3.237**
Many of these parameters have been amply explained for the ***Irradiance Map***. There are two new options though: ***Progressive path tracing (PPT)*** and ***Fly-through.***

Progressive path tracing - this method, an algorithm called ***PPT*** (**P**rogressive **P**ath **T**racing) is used for render calculation.
When a scene is rendered, effects like caustics, Global Illumination and so forth, are usually computed separately. By using ***Progressive path tracing*** the method by which renders are generated changes radically. ***VRay*** calculates the whole render in one single pass and the user sees the image appear in gradual steps, complete of GI, reflections and refractions, etc… It is a progressive method and the only parameter to set is ***Subdivs***. With this value one decides how many rays to use for the render. Once that threshold has been reached the rendering is blocked. Otherwise, if the quality is satisfactory, one can interrupt rendering, allowing the image which has been generated to be saved. The ***PPT*** only has one parameter to be set and can be considered the simplest of the methods described so far. As far as precision and quality are concerned it can be compared to the ***QMC***, but similarly to the ***quasi-Monte Carlo*** it is a very slow method, especially in indoor environments.



■ **Figure 3.238**
Some examples of ***PPT*** with various ***Subdivision*** values.

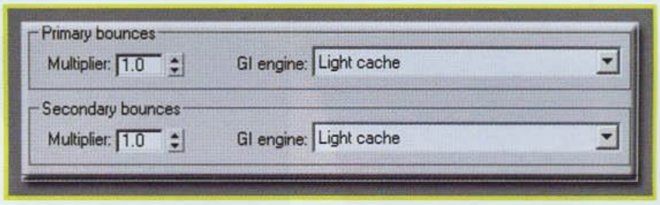
By doubling the number of subdivisions, rendering time quadruples. Also, in this example, already with ***Subdivs*** = 10,000 and 1h 20 min. elaboration, the render could almost be considered complete. From this value onwards the exponential increase in rendering time does allow improvements, such as cleaner image in the shadowy areas, but with such a great expense in terms of time that it is no longer convenient. With ***Subdivs*** = 40,000 the image has no flaws.



■ **Figure 3.239**
Noise curve.

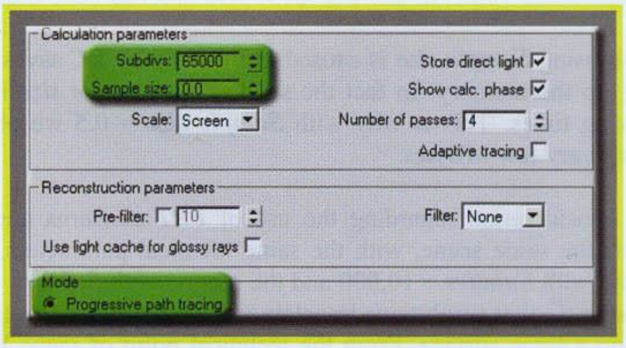
As we can see from the chart, after 3-4 hours of rendering, the loss of noise tends to slow down considerably and from 5 hours onwards the removal of what little noise is left becomes a long wait. It must be said that this chart is referred to a precise scene. If scenes are rich of details and complex shaders they may generate different "noise curves". In any case it is a generally correct concept that longer rendering times mean less noise.

Using ***PPT*** is very simple. Creating the scene, placing light sources and shaders and so on, are the usual procedures we have always applied before. In the moment we calculate the render, we only have to make settings for ***Primary*** and ***Secondary bounces*** for the ***Light Cache*** method.



■ **Figure 3.240**
Initial settings for ***PPT***.

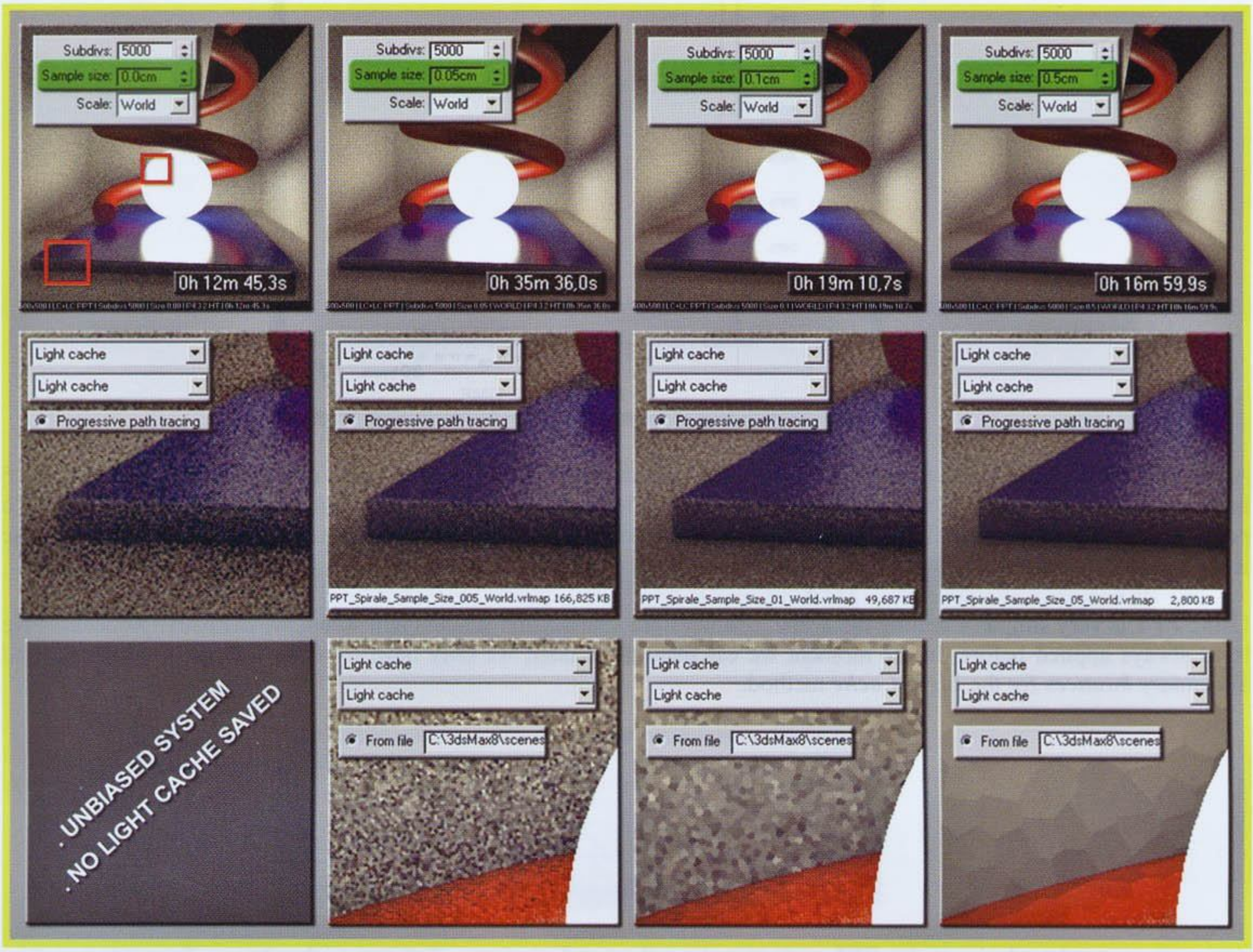
After, all we need to do is set the ***PPT*** method and set the number of rays that one wishes to use for rendering. The higher this value is, the higher the final quality is and also waiting time which reaches extremely high durations with ***Subdivs*** = 65,000, the maximum limit allowed by *VRay* in the current version. If ***Subdivs*** is set with medium-low values, like 500, 1,000 or 2,000, once *VRay* reaches that threshold it interrupts rendering and the render can be saved.
With ***PPT*** artifacts, multiple passes or imperfections caused by incorrect settings no longer exist and there is no need to seek the right conFiguretion for photons, the ***IM*** or the ***LC*** by trial and error. The only value that needs to be set is the number of ***Subdivisions***. For everything else, *VRay* will make decisions for you. Not even ***antialiasing***, ***Subdivs*** or ***glossy***, ***DOF*** or ***Area Shadows*** need to be set, as it is the ***PPT*** that chooses the correct value.



■ **Figure 3.241**
Subdivs, ***Sample size*** and ***Scale*** are the only parameters that are present in the ***VRay: Light cache*** and are of use for ***PPT***.

There is a second parameter that can influence the result of the ***PPT*** and consequently noise level: ***Sample size***.

So far we have always used value zero for this parameter. This means that ***PPT*** has been used as a pure ***unbiased*** engine, without approximation. By using ***Sample size*** > 0 a portion of the ***Light Cache*** is used for GI calculation. This makes ***PPT*** become a ***biased*** system. This gives us benefits for noise reduction but introduces the possibility that, for values which are too high, there can be ***light leaks*** at the base of objects, for instance at the bottom of walls. With high values larger ***LC*** samples are used. Small values decrease the bias, but more memory is required because there are more samples. If ***Sample size*** = 0.0 *VRay* does not use any cache, making it an ***unbiased*** solution.



■ **Figure 3.242**
Study regarding the variation of the ***Sample size*** parameter applied to ***GI*** calculation with the ***PPT*** method.

As ***Sample size*** increases, noise tends to drop, especially in areas with indirect lighting. What is more, by setting ***Sample size*** > 0 *VRay* introduces approximations which are typical of ***biased*** systems. Therefore at the end of a render *VRay* can save a GI which has been calculated up until that point, permitting one to not lose calculations carried out until that moment. When using ***Sample size***, *VRay* requires more RAM, the smaller the setting value is. ***Sample size***, just like ***LC***, depends on the ***Scale*** value. In this test the ***World*** unit has been used for the GI measurement, but one may also use the ***Screen*** unit.
The increased time due to using ***Sample size*** is caused exclusively by ***LC*** saves at the end of the ***PPT*** calculation and the operation of freeing up the memory. In fact the smaller the ***Sample size*** is set, the larger the ***LC*** is in terms of MBytes, with longer saving times. This is why with ***Sample size*** = 0.5 we obtain very low times, due to a saving process of a file containing very few MBytes.

Another very important consideration regarding the use of ***PPT*** concerns output resolution. In the next test some renders are carried out on the same scene, with the same identical parameters, but at different resolutions. The first series of tests is carried out with ***Subdivs*** = 10,000 and the second with ***Subdivs*** = 20,000. It can be observed how, with the same number of ***Subdivs***, higher resolutions translate into higher noise levels. This is because the samples used by the ***PPT*** are distributed on a large surface. From the technical point of view, by doubling the resolution, in order to obtain renders of the same quality one must multiply the ***Subdivs*** by four. This means that time needed for rendering is also four times as much.

■ **Figure 3.243**
Disturbance increases in images with changing resolutions.
- - - DVD - - -

NOTES

- The sampling of the image, or ***antialiasing***, is ignored with ***PPT*** use, both in ***Fixed***, ***Adaptive QMC*** and ***Adaptive subdivision*** modes. Image supersampling of the image is carried out and calculated automatically by the ***unbiased*** method.

- The only ***antialiasing*** parameters considered are filters. The so-called "hard" ones, like ***Mitchell-Netravali*** or ***Catmull-Rom*** can then generate more noise compared to smoother filters, such as ***Blend***, also softer ones can take more time for calculation. It is advisable to leave them disabled.

- The only ***QMC*** sampler parameters considered with this method are ***Adaptive amount*** and ***Time-independent***. It is best not to set ***Adaptive amount*** = 0 when using ***Path tracing***.

- When using ***PPT***, global illumination does not have limitations on the number of bounces. Those dedicated to reflections/refractions alone, instead, are managed by the single parameters of the shaders or otherwise globally by the parameter in the ***VRay: Global switches*** rollout.

In this version of *VRay*, the ***PPT*** can "only" generate 2^32 paths. In fact the ***Subdivs*** spinner of the ***Light Cache*** is purposefully limited to 65,000, which allows one to obtain 65,000^2 = 4,225,000.000 rays, distributed throughout the image. This is why for renders with high resolutions like 2000 x 2000 pixels, it is not impossible for the ***PPT*** to generate images completely free of noise, because for each pixel there are at the most 1,000 rays available. If this is the case it is advisable to use the traditional ***QMC GI*** method.

Single frame - in this mode the ***LC*** is computed for every single frame. Once ***LC*** calculation is complete, the map can be manually saved by clicking the ***Save to file*** button. One can then specify a destination for the save.

Fly-throught - this method is useful in animations where only the camera changes position or moves. Such animations are called ***Fly-through*** animations. In this mode, *VRay* calculates ***Light Cache*** for the whole animation, using the number of frames in the Time Line as time length for the calculation. For this type of animations it is advisable to use the ***World*** unit as a scale value so that samples always have the same size, wether they are close to the viewpoint or far away. This avoids flickering problems one might otherwise encounter. If during ***Light Cache*** calculation the ***Show calc. phase*** option is activated, *VRay* shows, in the ***Frame buffer***, a "cloud" of rather baffling points. *VRay* is actually calculating the ***LC*** for the whole animation. Once the whole calculation is over and the map has been saved, the GI will have been perfectly calculated and we can observe this from any position we choose to take.

From file - this option permits the ***Light Cache*** to be loaded into the memory. Remember that in the ***.vrlmap*** file, neither ***Pre-filter***, ***Nearest*** or ***Fixed*** filters are contained. ***Pre-filter*** is calculated just before rendering in the ***Frame buffer***. So it is possible to act upon these parameters also after having calculated the ***Light Cache***, without having to re-calculate it every time changes are applied to filters.

On render end

The last part of the ***VRay: Light cache*** rollout is the ***On render end*** section. These commands allow one to manage the ***LC*** after the final render has been carried out. The generated map can be automatically erased, held in the memory, saved to a file or, automatically loaded into the ***From file*** box.

■ **Figure 2.244**
These options allow ***LC*** to be managed at the end of rendering.

Don't delete - this is the default option. *VRay* holds the ***LC*** in the memory at the end of calculation, making it available for manual saves. If it is disabled, after rendering the ***.vrmap*** is automatically erased, making it impossible to save, either manually or automatically.

Auto save - if activated, this parameter allows users to insert a save destination.

Switch to save map - this parameter is available only when ***Auto save*** is active. Once the render is over, *VRay* loads the ***LC*** file in ***From file*** mode automatically.

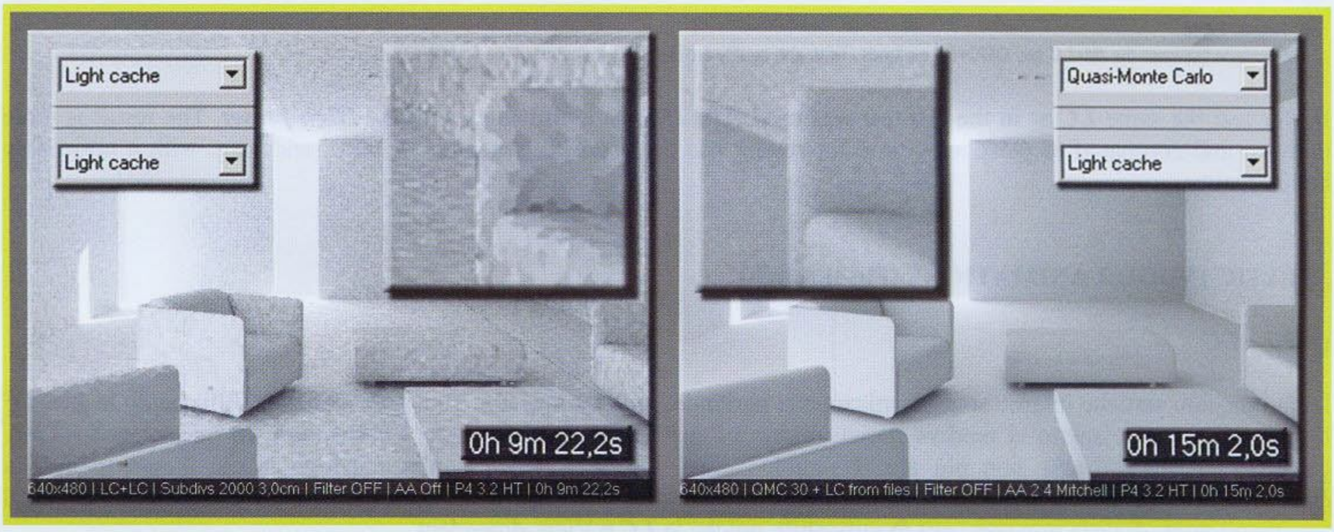
TYPES OF USE

The *Light Cache* is a method which in most cases allows one to obtain high results with little effort, from the point of view of hardware and technique. Setup is simple, calculations are fast and these qualities make *Light Cache* one of *VRay*'s strong points. 99% of the time *Light Cache* is used for *Secondary bounces*. Apart from *PPT*, it is difficult to obtain clean results if one uses *LC* also in *Primary bounces*.

. STATIC INDOOR RENDERS

QMC + LC

This pair can easily be used in this type of project. Results are excellent, thanks to *QMC* as a primary engine, whereas *Light Cache* assures fast rendering, making the use of *QMC* competitive.
In the case examined the render has been carried out in two phases, purely with the purpose of learning from it. Firstly the *LC* map was calculated with the *LC+LC* method. One saved and loaded into the memory, the *QMC+LC* method was used. At this point all that was left to do was to set the number of *Subdivs* for *QMC*. A typical feature of renders obtained with this method is the classic grainy appearance of the image due to *QMC*, which became fashionable after the arrival of Maxwell, which reminds one vaguely of the disturbance typical of photographic film and gives the render an extra feeling of photorealism.



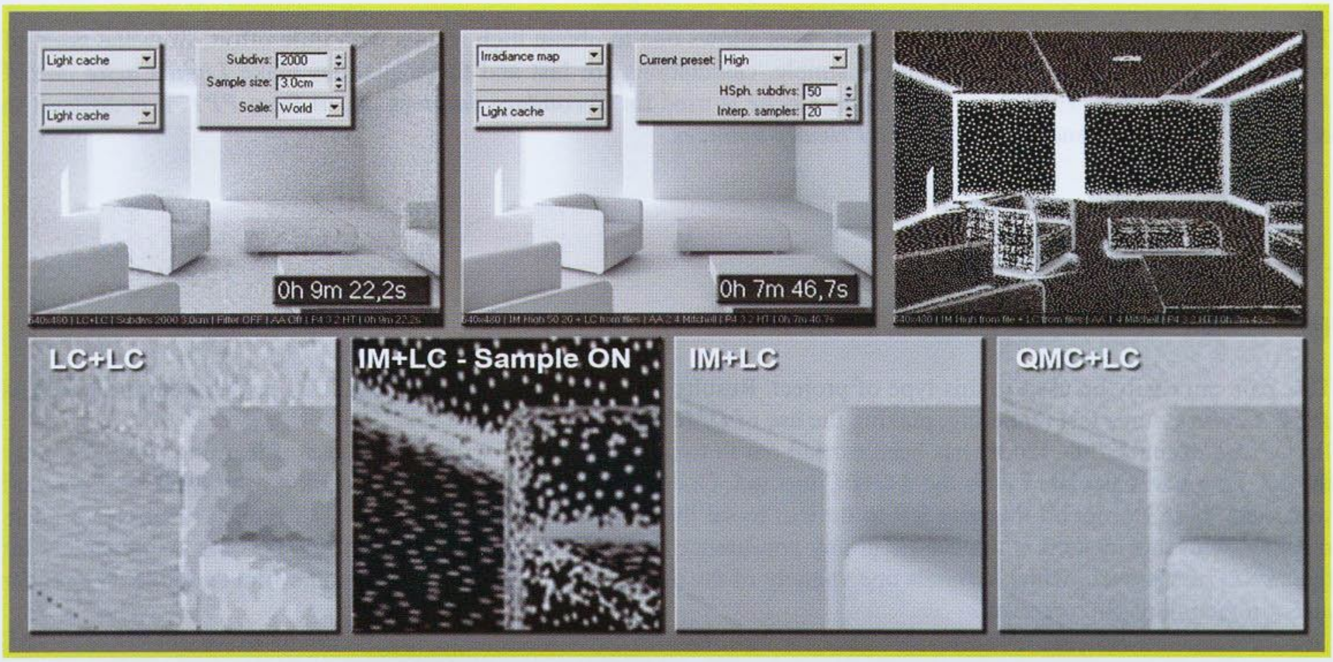
■ **Figure 3.245**
The ***QMC+LC*** method is extremely precise but also slow.

In this example *Light Cache*, set to medium values, took around 10 min. for the complete calculation. After the final image was rendered using *QMC* with the freshly-made *LC*. This frame required about **25 min**. To reduce rendering time it would have been sufficient to reduce the *LC* quality slightly. In order to obtain satisfactory results with indoor scenes with *QMC*, 30 subdivisions are the necessary minimum and one cannot hope to go much lower than this value. At the end a medium-high *antialiasing* filter was applied, which contributed towards the production of a quality render, but also an increase in rendering time.

IM + LC

This is the method which is used most within *VRay*, because of its high quality and extremely fast calculations. Thanks to the combination of these two biased methods, images are extremely clean and fast to render. The possibility of customizing the setup is also a handy feature. One can generate fast renders or very precise and accurate ones, which require additional resources. The time needed to setup the *Primary bounces* is greater than when using *QMC*. Using these two methods one may proceed to GI calculation in one pass or multiple passes. One can calculate *LC+LC*, save it, load it and change the setup into *IM+LC* and compute the final image, or both at the same time. It is advisable to calculate them separately, at least for preliminary tests, so that once the correct setup has been found for *LC* one can concentrate on the *IM*.

Tips: once the ***IM*** has been computed using the previously-calculated ***LC*** as a ***Secondary bounce***, one can set ***Secondary bounces*** to ***None***. This is possible thanks to the fact that the ***IM*** contains information concerning the ***LC*** and therefore saving a lot of energy. But this procedure is not valid if the ***LC*** is used for glossy calculations. In this case it is necessary to use it.



■ **Figure 3.246**
Comparison between the ***IM+LC*** and ***QMC+LC*** methods. Time and quality are definitely in favour of the first pair.

In this test the same ***LC*** as in the last test was used. The ***IM*** has high values with the ***High preset***. The result is quite uniform and compared to ***QMC*** use, time for render has been halved, reaching 17 min. rendering, with a noticeably better quality.

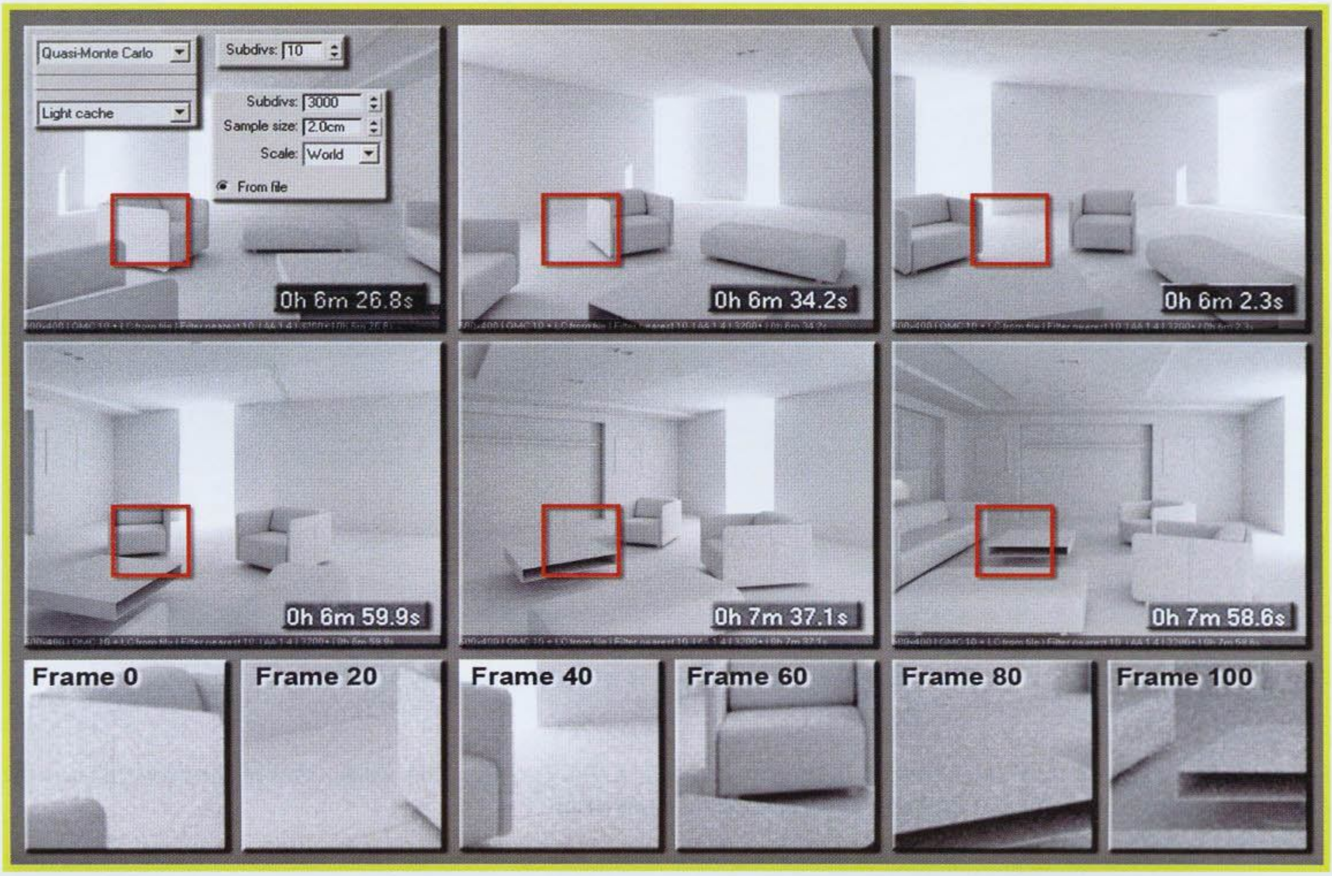
. STATIC INDOOR ANIMATIONS (Fly-through)

QMC + LC

Le animazioni statiche interne sono, tecnicamente parlando, molto simili alle immagini statiche. Utilizzando la ***LC+LC*** l'unica variazione rispetto ai renders statici interni è l'attivazione del parametro ***Fly-through***. In questo modo *VRay* calcolerà la ***Light Cache*** non più per il singolo frame, ma per l'intera dell'animazione la cui durata è definita dall'impostazione della Time Line di 3ds Max, computandola per tutto il percorso. Una volta ottenuta la mappa, il tragitto e i punti di vista non sono più modificabili, poiché la ***LC*** è View-dipendent.

■ **Figure 3.247**
Settings for the calculation of the ***.vrlmap*** file during a ***Fly-through*** animation with an AMD 3200+.

As the length of the animation is 200 frames, the ***LC*** requires about 10 sec. for each frame (35 min. 44 sec./200 frames). Once ***LC*** calculation is complete, the generated map is loaded into the memory and used for Secondary bounces. For the ***Primary bounces QMC*** is set. Because of the ***QMC***'s nature, GI has to be calculated for each frame, even if there is no change in lighting for this type of animation. With an average of 6 min. 30 sec. for the calculation of the final render, as well as the 10 sec. needed for ***LC*** calculation, each frame requires 6 min. 40 sec. This means 22 hours of rendering (6 min 40 sec. x 200 frames)!



■ **Figure 3.248**
Some examples of frames taken from the indoor static animation, generated via the ***QMC+LC*** method.
--- DVD ANIMATION ---

In this example, fulfilled with a rather out-of-date PC, two situations can be emphasized. The first is the strong presence of noise throughout the whole animation. The second concerns rendering time which, for a 500 x 400 pixel resolution in a relatively simple scene, is very high. The reason is the use of ***QMC*** which is not appropriate for this type of project, although it guarantees high quality results.

IM + LC

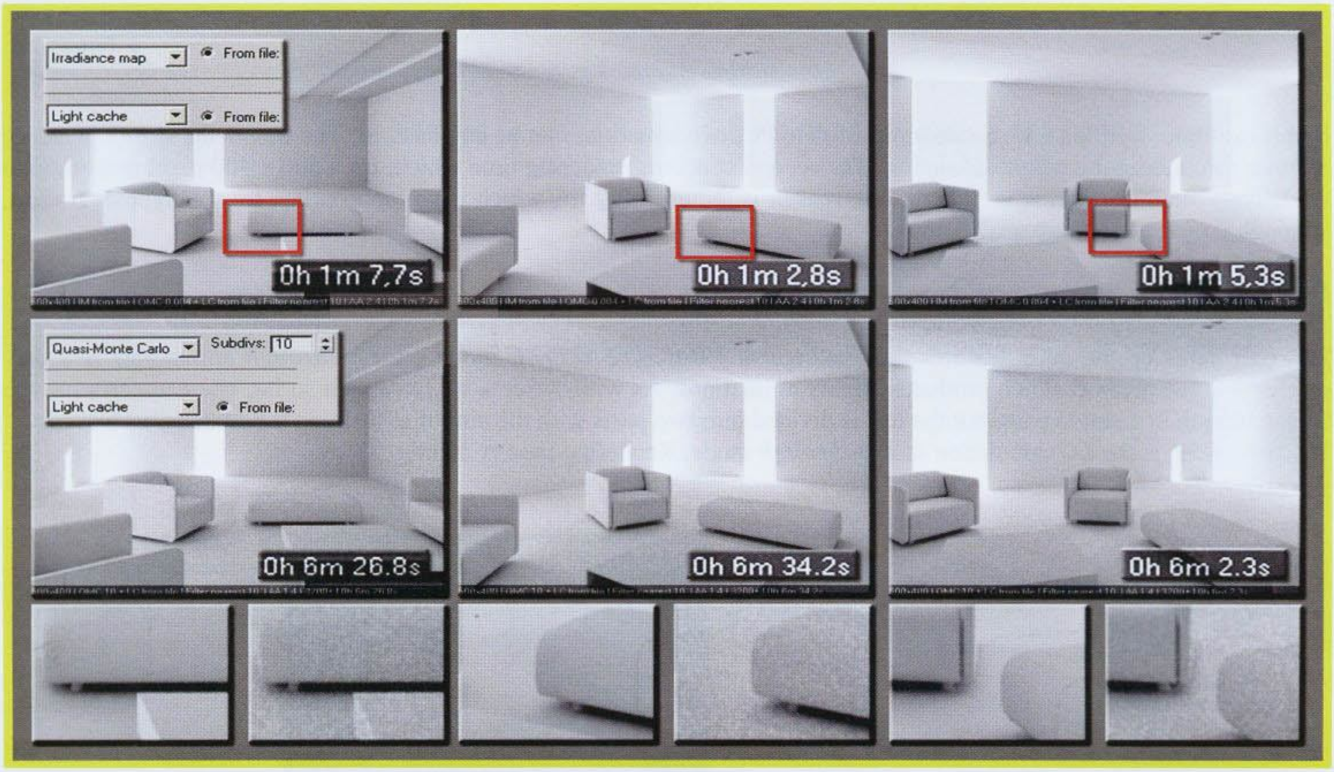
This pair sums up the main features that distinguish *VRay*. The ***IM+LC*** mode for ***Fly-through*** animations is a very solid and quick method which produces quality animations. As with systems we have seen so far, one can carry out the GI calculation in a single pass, but the test is divided into two parts with the aim of explaining it better.
The first phase is the ***LC*** calculation in ***Fly-through*** mode. Rendering time is 21 min. 28 sec.

■ **Figure 3.249**
Calculation of the ***LC*** for ***Fly-through*** animations using a P4 3.2.

Once loaded in the memory, the ***IM*** is used for ***Primary bounces***. A frame interval is established within which the ***IM*** is to be calculated via the ***Incremental add to current map*** mode. The ***IM*** is calculated and after, saved.

■ **Figure 3.250**
The 20 frames used for incremental calculation of the ***IM***, which by the end weighs 16 MByte. The ***LC*** instead weighs 32 MByte.
--- DVD ANIMATION ---

For the ***IM*** to be completely calculated, 4,980 sec. were needed, that is, 1h 23 min. At the end of these two phases the ***IM*** and ***LC*** maps are loaded into the memory, ready for final rendering.



■ **Figure 3.251**
Once the ***IM*** has been generated and loaded into the memory, rendering speed depends only on shader characteristics and the lighting used in the scene. In our case, having used a shader with one color only, rendering time is very low.

As one would expect, with *IM+LC* this scene required much less time to render than *QMC+LC*, with a higher quality level. There is no noise and there are no flickering problems, because of the medium-high values used for *IM* calculation (*Max Rate* = -3 *Max Rate* = 0 -0.3 0.3 0.1 - *HSph.* = 80 *Interp* = 20).
The average per frame for this last phase is around 1 min. 5 sec. This means about 13,000 sec. (65 sec. per frame x 200 frames) which gives us 3h 30 min.

. 21 min. - *LC* calculation
. 1 hour 23 min. - *IM* calculation
. 3 hours 30 min. - final rendering calculation

5 hours 14 min.

Which means 1min. 34 sec. per frame.

For this test a P4 3.2 HT was used, which is 25% faster than an AMD 3200+ used for the *QMC+LC*. In order to compare results one must multiply 1 min. 34 sec. x 1.25. Final result is therefore:

. 1 min. 57 sec. - *IM+LC* method
. 6 min. 40 sec. - *QMC+LC* method

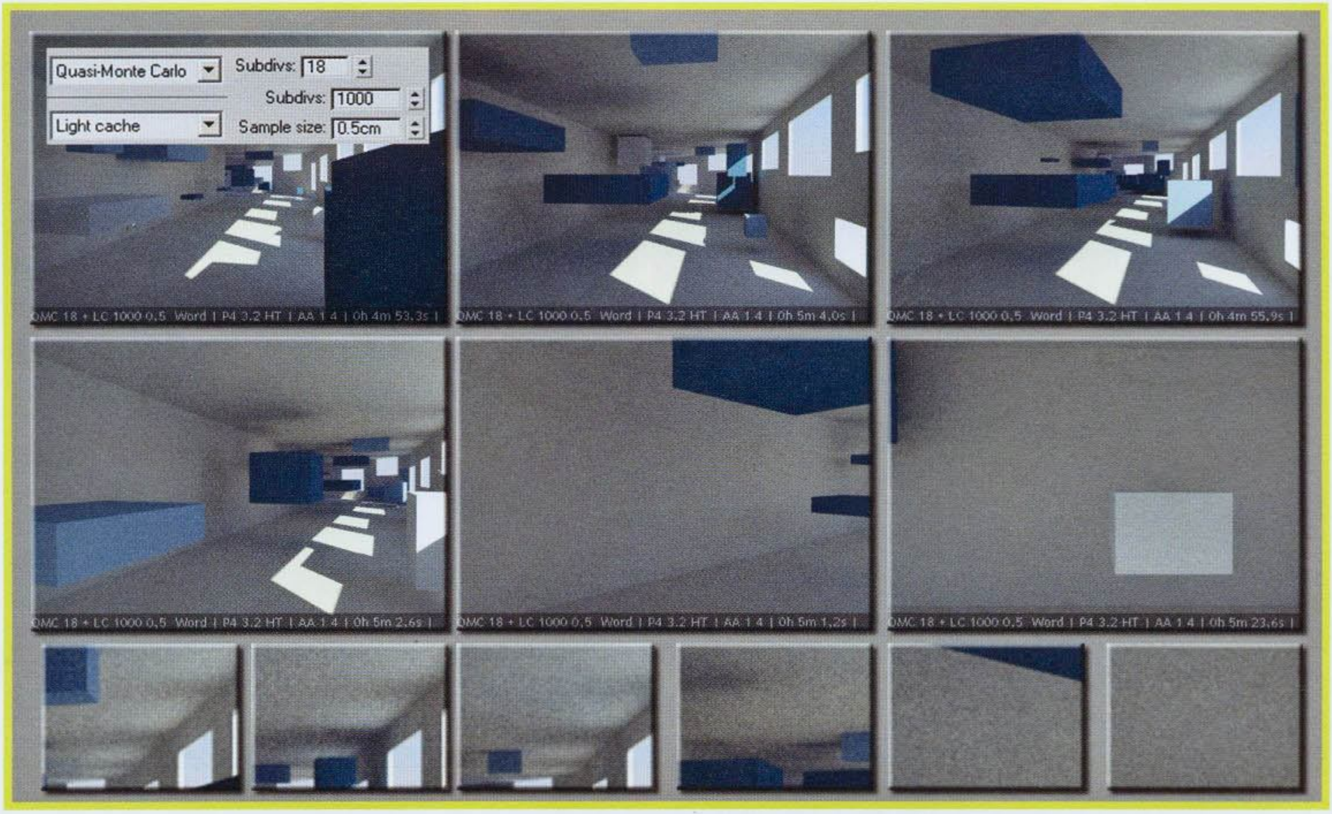
With an obvious difference in quality. In conclusion: approximative so-called biased methods, are unbeatable for this type of project!

. DYNAMIC INDOOR ANIMATIONS

As with all dynamic animations, the fact of not being able to save the GI makes ***biased*** systems less useful and open possibilities for the use of ***unbiased*** methods, such as *QMC*. But according to the scene being created, one can also use biased methods beneficially, as they still provide clear and good quality animations with reasonable times.

QMC + LC

The presence of noise is the main problem of this method, as well as slow rendering.

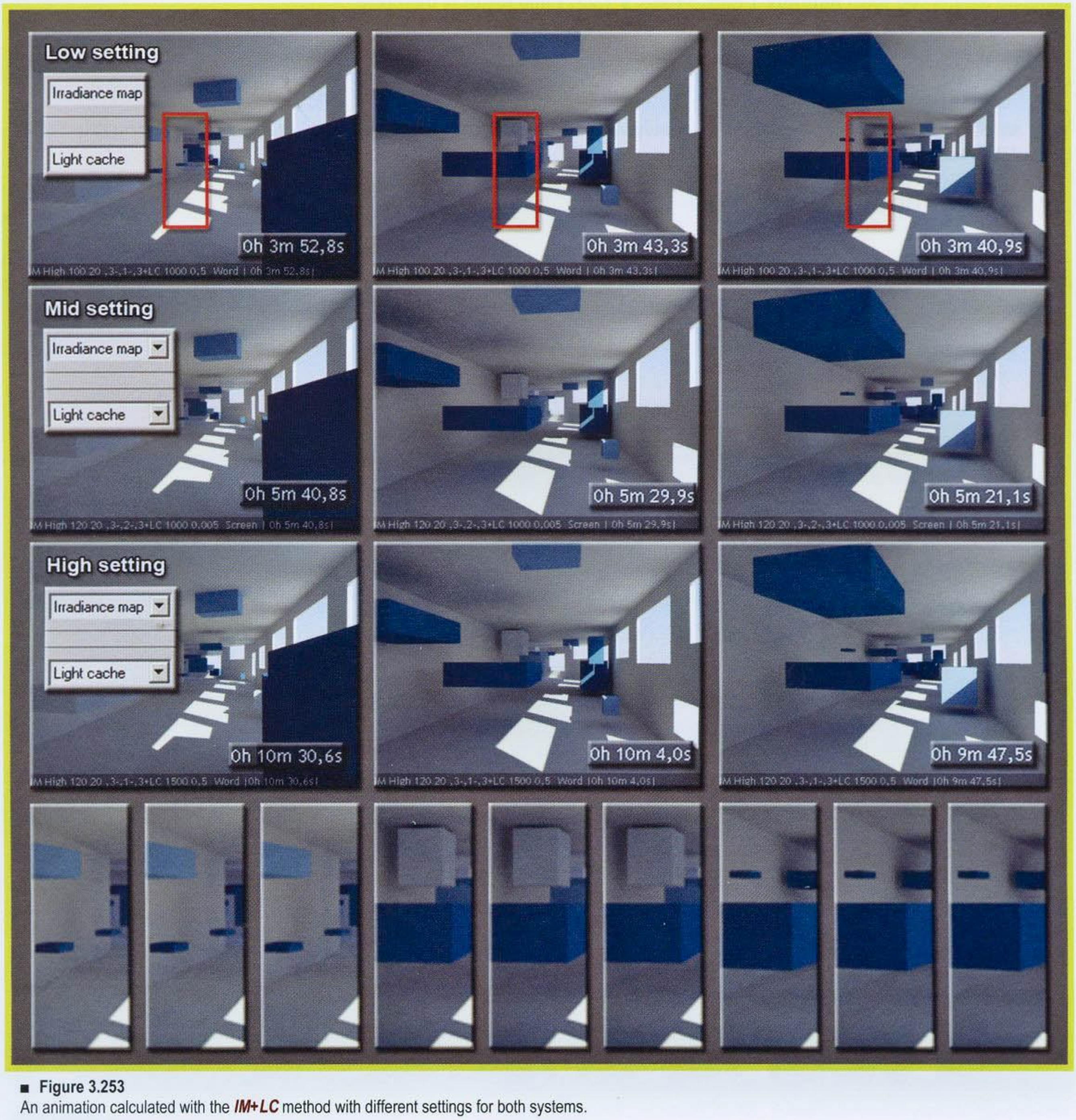


■ **Figure 3.252**
Images are very detailed, shadows are well-defined and there is no flickering. Noise can only be eliminated by increasing ***QMC Subdivs***. With consequent increase in rendering time.
--- DVD ANIMATION ---

The strong point in ***QMC*** animation lies in its ability to eliminate flickering problems. If there were any, certainly the cause would be found in an incorrect ***Light Cache*** setting, with insufficient subdivisions, excessive ***Sample size*** setting or filters.
In this example the animation is made up of 400 frames. ***Light Cache*** cannot be pre-calculated and each frame must be elaborated as a single static render. This is why time per frame has been kept as low as possible, taking care to be on the alert for signs of flickering. This resulted in a rendering time of about 5 min. per frame, which gives a total of 33 hours and 20 minutes for the whole sequence.

IM + LC

Just like with the ***QMC+LC*** method, also for this system we must calculate both ***IM*** and ***LC*** for each frame. The main problem with this pair is flickering, which is caused by a slight difference in lighting between frames in areas which are particularly hard for GI calculation, such as edges or small surfaces. The only remedy for this problem is to use very high calculation parameters for ***LC*** and ***IM***.



■ **Figure 3.253**
An animation calculated with the ***IM+LC*** method with different settings for both systems.

In this example, settings were adjusted in the following order: medium, medium-high and high for both the ***IM*** and the ***LC***.

. 1° TEST: IM High PRESET 100 Subdivs 20 Interp. - LC 1,000 Subdivision
……………3 min. 50 sec. … per frame.
...**25 hours 33 min.** ………... for the whole animation.

After the ***IM*** quality was increased, bringing ***Subdivs*** to 120.

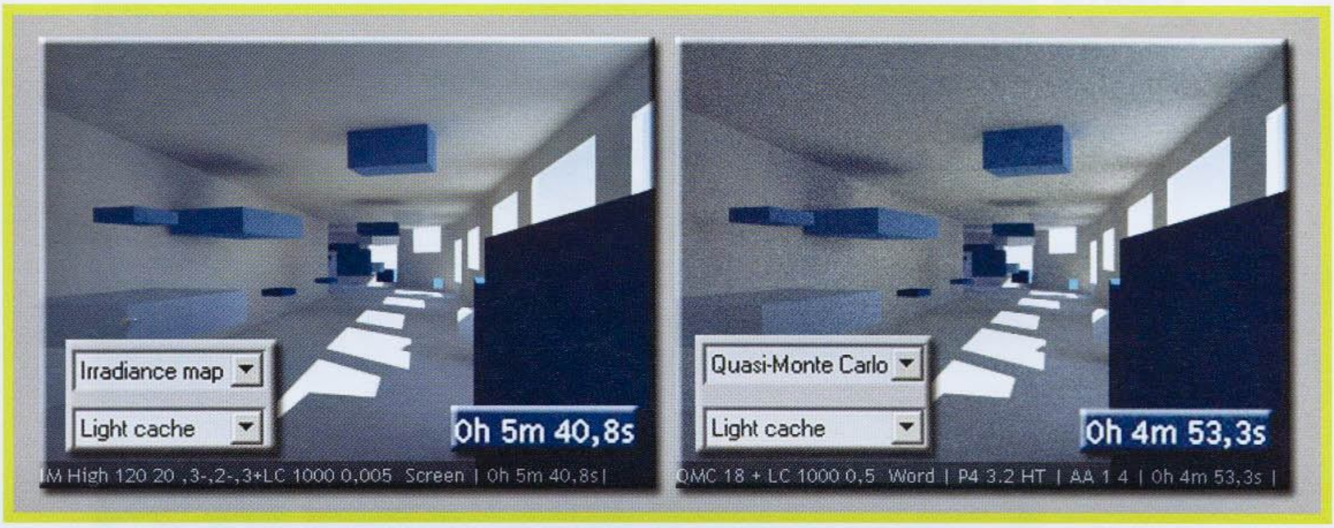
. 2° TEST: IM High PRESET 120 Subdivs 20 Interp. - LC 1,000 Subdivision.
……………..5 min. 30 sec … per frame.
…**36 hours 40 min.** ………….. for the whole animation.

Lastly ***subdivisions*** were increased for the ***Light Cache***, bringing it from 1,000 to 1,500:

. 3° TEST: IM High PRESET 120 Subdivs 20 Interp. - LC 1,500 Subdivision.
……………10 min. … per frame,
….**66 hours 40 min.** … for the whole animation.

The images obtained, when compared, seem identical. In fact the difference is minimal. But whilst playing the animation, these little differences can cause flickering.

Lastly, the comparison between the ***QMC+LC*** and the ***IM+LC*** method.



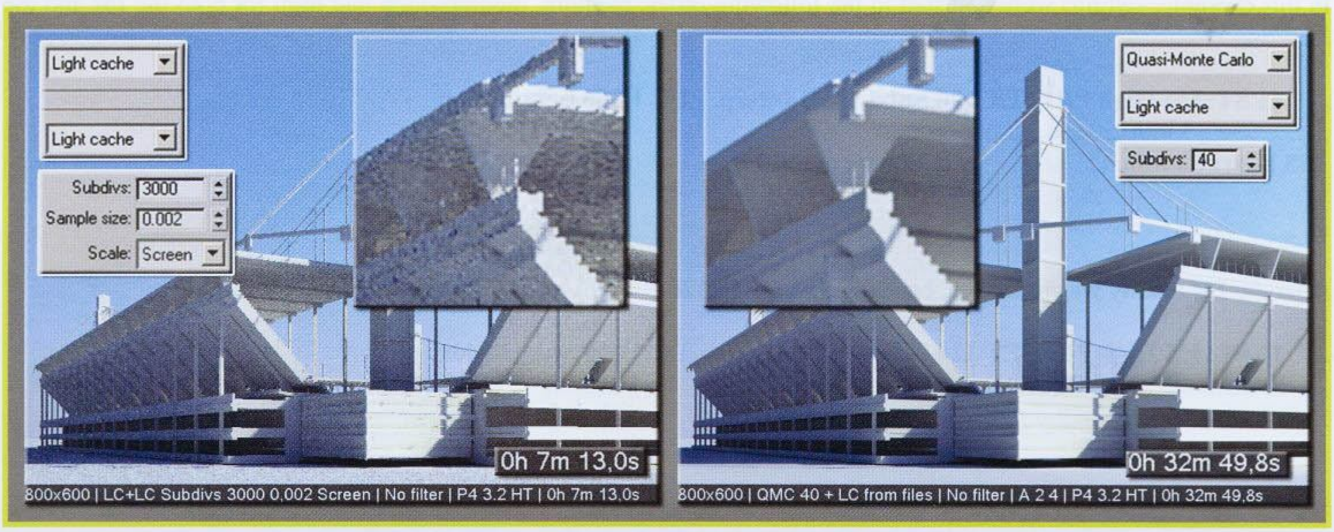
■ **Figure 3.254**
The presence of noise in the ***QMC*** is evident: there is no solution except increasing subdivisions. Using ***IM*** instead, results are more than satisfactory.

There is no comparison. In this scene, using ***IM*** provides noticeable improvements to the whole animation, making it more uniform and cleaner.

. STATIC OUTDOOR RENDERS

For outdoors, which do not involve secondary bounces nearly as much, the choice between ***QMC+LC*** or ***IM+LC*** is more difficult. On one hand, one can choose a render which is not at risk from artifacts as well as quick setup times and superior detail, on the other hand, the alternative allows GI to be saved, renders to be re-used at higher resolutions, better uniformity and much lower rendering times. The choice between these two methods is also decided on the basis of other factors, such as the use of ***glossy*** shaders, ***DOF, Motion blur*** or ***Area Shadows***: in indoor environments ***IM*** may be the best choice for these, instead the same is not necessarily true for outdoor scenes.

QMC + LC

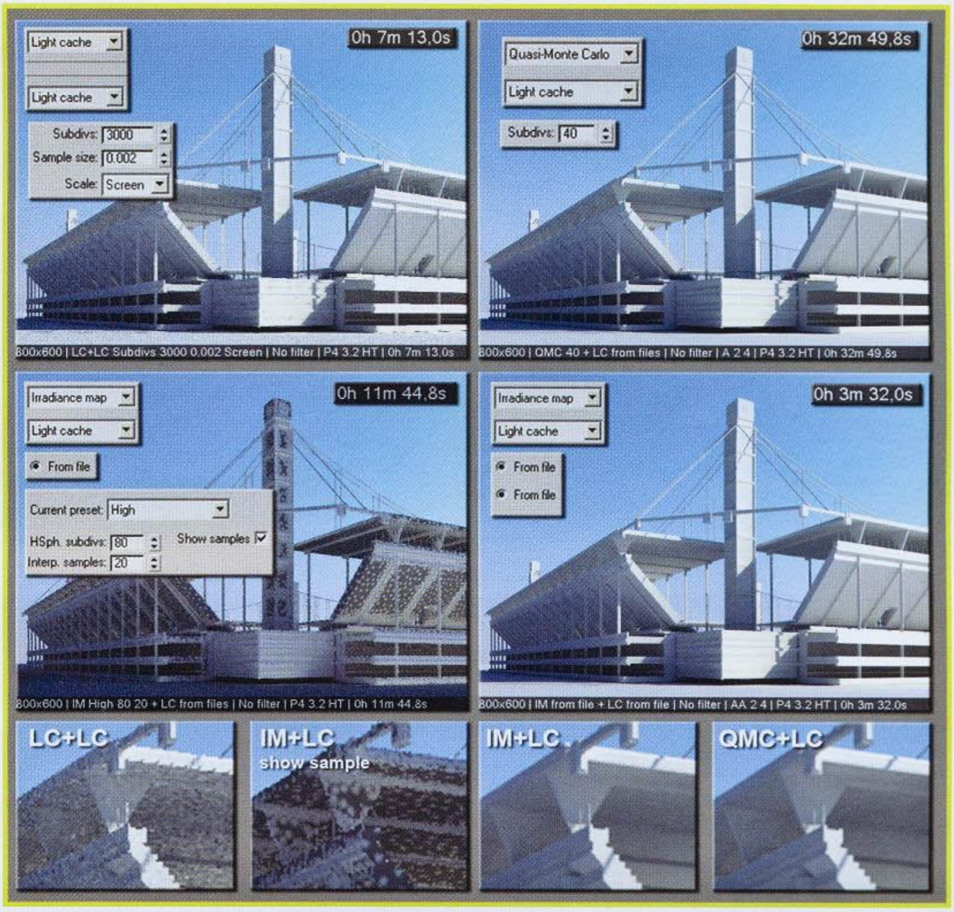


■ **Figure 3.255**
An outdoor rendering carried out with the ***QMC+LC*** method in two phases.

As in the previous examples, GI calculation has been divided into two phases for learning purposes. The first of the two concerns ***LC*** calculation at medium values. The time required was 7 min. 13 sec. After the ***LC*** was loaded into the memory in order to generate the final image with ***QMC*** as the primary engine. Also ***QMC antialiasing*** was activated with ***Min subdivs*** = 2 and ***Max subdivs*** = 4. A good 32 min. were required for the full calculation at a resolution of 800 x 600 pixels. Adding the ***LC***, the generation of this image required about 40 min, with a surprisingly high level of detail and clearness for a system like ***QMC***.

IM + LC

■ **Figure 3.256**
Comparison between the ***IM+LC*** method and the ***QMC+LC*** method in an outdoor static render. With the ***Irradiance Map*** one can obtain very clean and precise results.



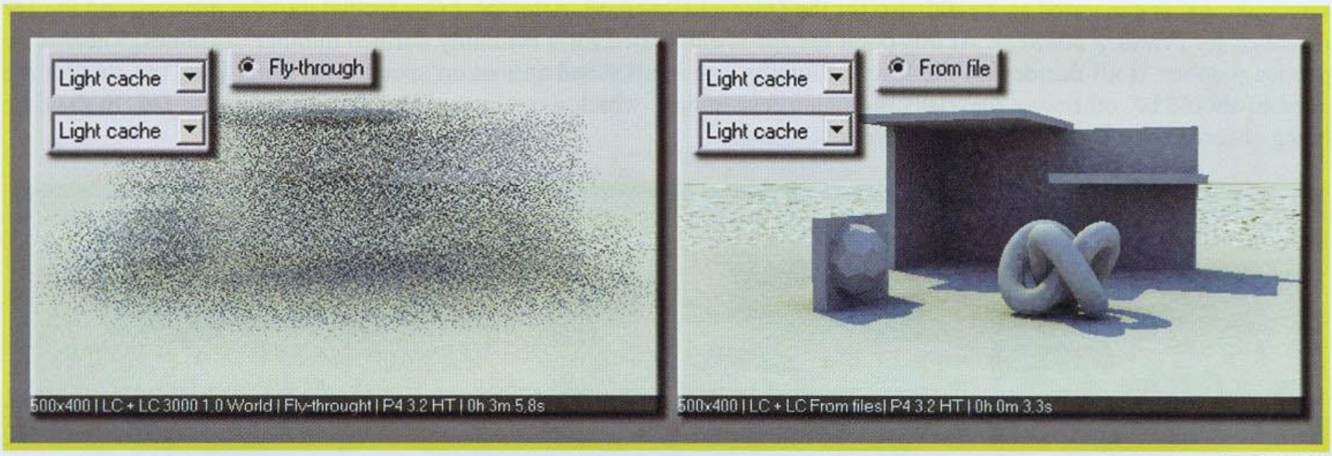
In little more than 20 min., half the time taken by the ***QMC+LC*** method, thanks to the use of the ***Irradiance Map***, it was possible to obtain cleaner and more precise renders. Also, thanks to the fact of being able to save both the ***LC*** and the ***IM***, one can render at a greater resolution than the one used in calculation, therefore employing very few minutes.

. Dynami (Fly-through)

In using an outdoor animation, we will analyze assets and disadvantages of the main two methods which can be used with the ***LC***, namely ***QMC*** and ***IM***.

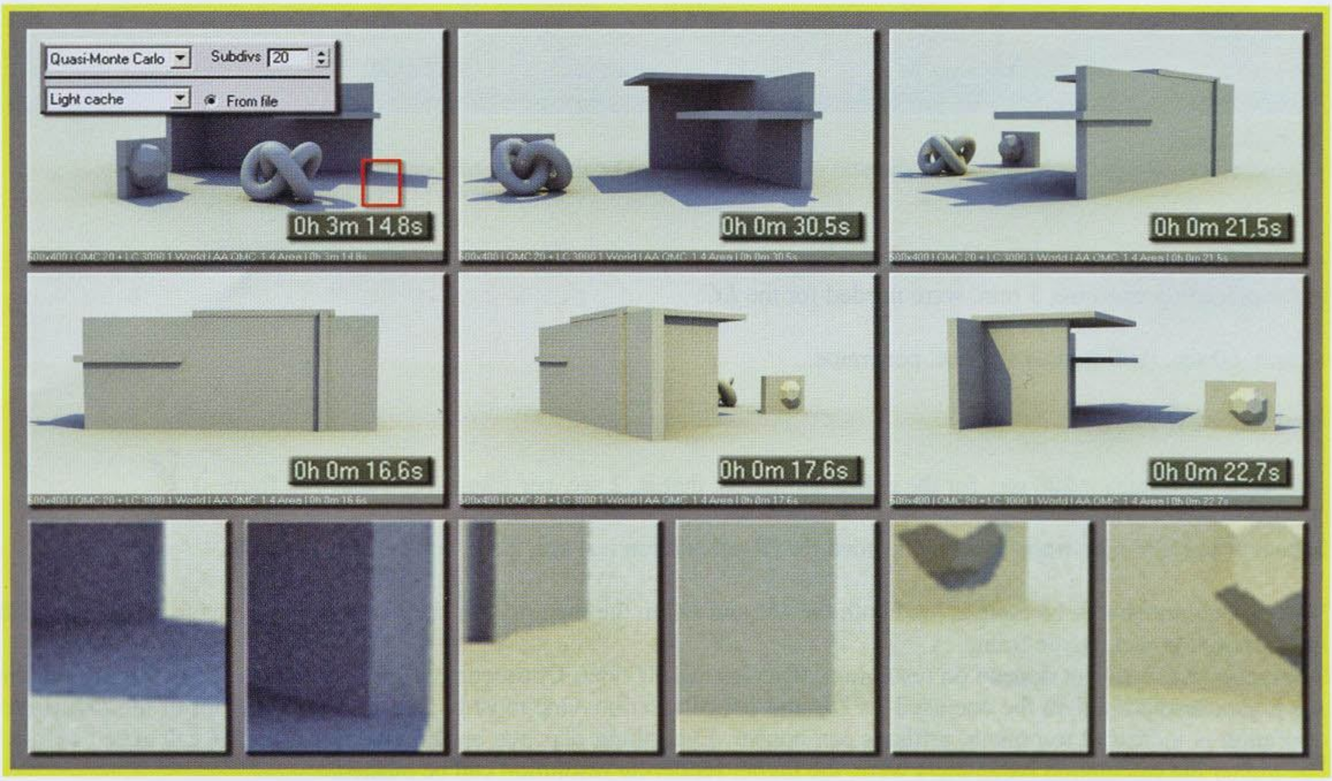
QMC + LC

The first step is to calculate the ***LC*** via ***Fly-through*** mode. ***VRay*** will then calculate the GI for the whole animation through the ***LC***. When this happens, in the viewport one can observe a cloud of samples which go to make up the ***.vlmap*** file. Once the map is calculated, it is loaded into the memory. At this point, by rendering any frame in the animation, one can observe the correctly computed ***LC***.



■ **Figure 3.257**
Cloud of ***samples*** generated by the ***Light Cache*** in ***Fly-Through*** mode and rendering of a single frame taken from the previously saved map.

Once the ***LC*** is charged in the memory with ***QMC*** as ***Primary bounce*** engine, the final animation is carried out. In this type of animation, the only object which moves is the camera and therefore one can avoid calculating the GI in areas where it has already been calculated. Unfortunately, via ***QMC***, this is impossible, and rendering time increases.



■ **Figure 3.258**
6 of the 100 frames which make up the animation, calculated with the ***QMC+LC*** method.

In the exercise we have just seen, the ***QMC*** has medium settings, with ***Subdivs*** = 20. The calculation is carried out with ***antialiasing Adaptive QMC***, ***Min subdivs*** = 1 and ***Max subdivs*** = 4. This led to a rendering time of roughly 20 sec. per frame allowing the whole 100 frame-long animation to be finished in about 33 min. There are however evident noise problems, but the sequence is flickering-free.

IM + LC

For calculating the animation with the use of ***IM+LC*** a few steps are necessary. Much like for static animations where the ***Irradiance Map*** is involved, it is not possible to calculate the animation all in one go, but the user's manual adjustment is needed for changing some parameters, such as the ***Every Nth Frame*** setting in the ***Common*** window, loading the ***IM*** , and so on. Technically it would be possible to calculate both the ***IM*** and the ***LC*** together but here they have been fulfilled separately so as to illustrate the process better.

Firstly the ***LC*** is generated as we did also for the ***QMC+LC*** method. After the ***LC*** is loaded into the memory and the ***IM*** is loaded for ***Primary bounces***. At this point the ***IM*** is elaborated for a manually defined number of frames. There is no precise number. It all decides on the level of quality one is after and animation speed. The faster it is, the smaller the interval should be, otherwise one risks missing a few frames, which would mean having an incomplete ***IM***. In the case being observed, the ***IM*** was calculated every 10 frames.

■ **Figure 3.259**
Calculation of the ***IM*** and ***LC*** for an outdoor static animation. At the end of elaboration two files are obtained: ***.vrmap*** and ***.vlmap***.

In the preceding exercise, 3 min. were needed for the ***LC***:

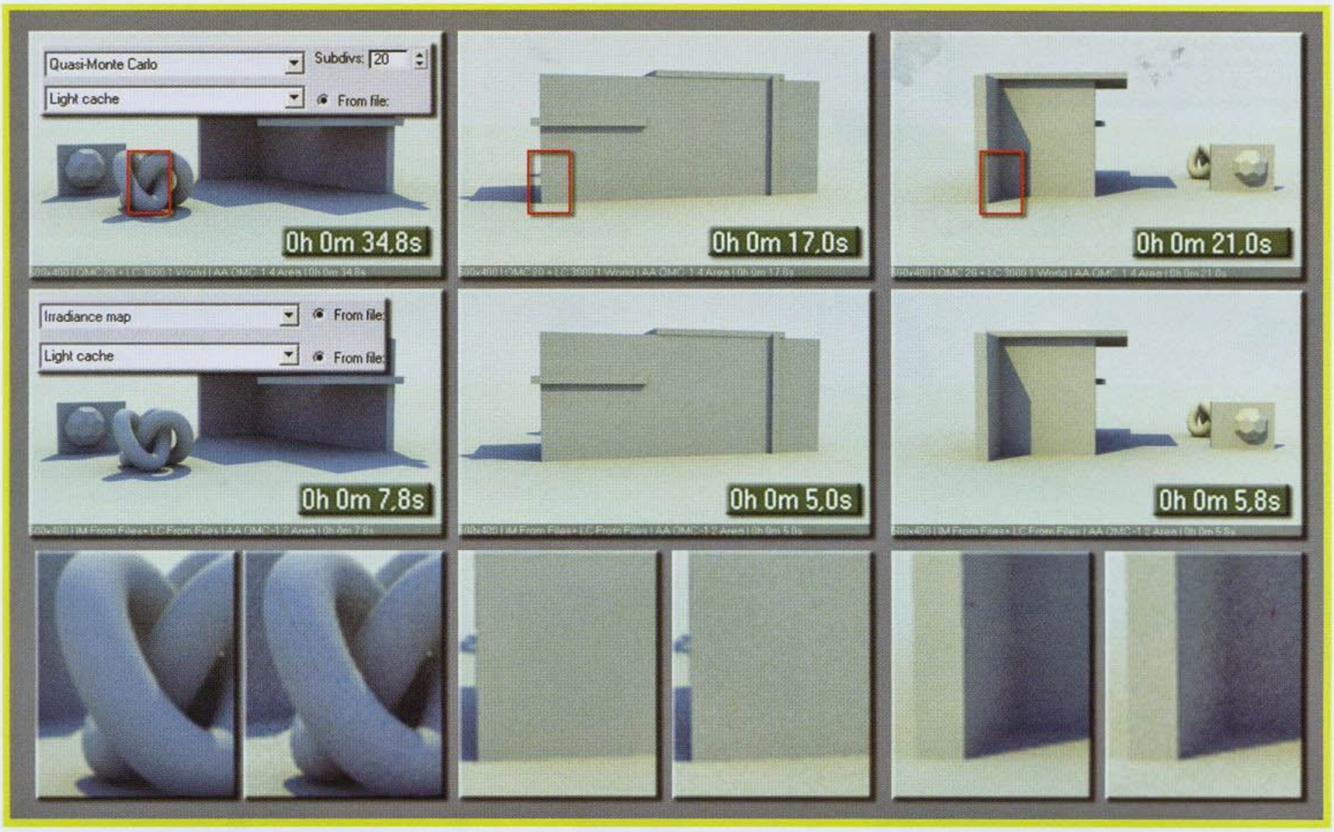
. 3 min. 60 sec. /100 frames = 2 sec. per frame.

instead for ***IM*** :

. 20 sec. x 10 frames = 200 sec. for the entire animation, that is, 2 sec. per frame. (200 sec./100 frames)

In conclusion, for each frame the time needed for GI calculation is 4 sec, 2 sec. for ***LC*** and 2 for ***IM***.

Now all that remains to be done is load both the ***IM*** and ***LC*** in the memory via the ***From files*** command and launch the final render for each single frame.
Remember: ***LC*** does not depend on resolution, whereas the ***IM*** does. Consequently it is possible to render the animation at a higher resolution than the one used for ***IM*** and ***LC*** calculation. One must be aware of the behavior of the ***IM***. If the resolution is increased too much, artifacts can appear. Everything depends on the quality of the ***IM*** calculated before. The higher the calculation parameters were, the higher rendering resolution can be pushed.



■ **Figure 3.260**
Comparison between the ***QMC+LC*** method and the ***IM+LC*** method in an outdoor static animation.

Time for rendering per frame with *IM+LC* is about 6 seconds. Plus 4 seconds for GI calculation gives an average time of 10 seconds. Instead the *QMC+LC* method gives an average of 20 seconds. Rendering time has been halved, returning better quality with no noise. A disadvantage for the *IM+LC* method is setup time but this is made up for by its fast rendering time.

QMC+LC = 20 sec. per frame

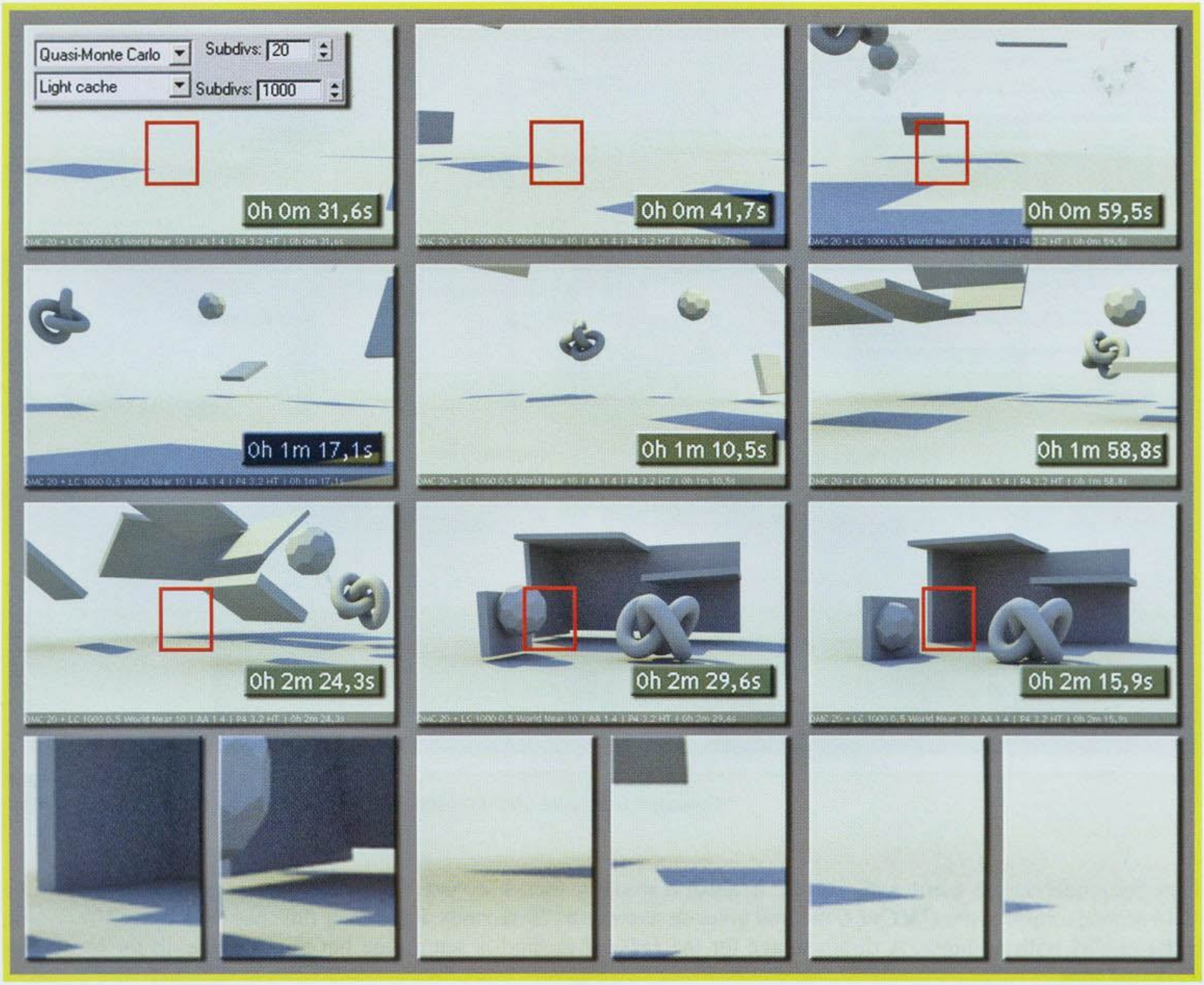
IM+LC = 10 sec. per frame

. DYNAMIC OUTDOOR ANIMATIONS

In this type of scene the GI must be calculated for every frame, so it is not possible to save it and re-use it.

QMC + LC

When using the *QMC* for ***Primary bounces*** we are certain to eliminate any type of flickering, although there will almost certainly be problems with noise and high rendering times. Unfortunately the more noise is lowered, the more time is required exponentially, therefore one must try to obtain the cleanest possible render according to the time available and the hardware one has. In any case, outdoor scenes are not as prone to noise as indoor ones, so it is possible to use *QMC* with high values.

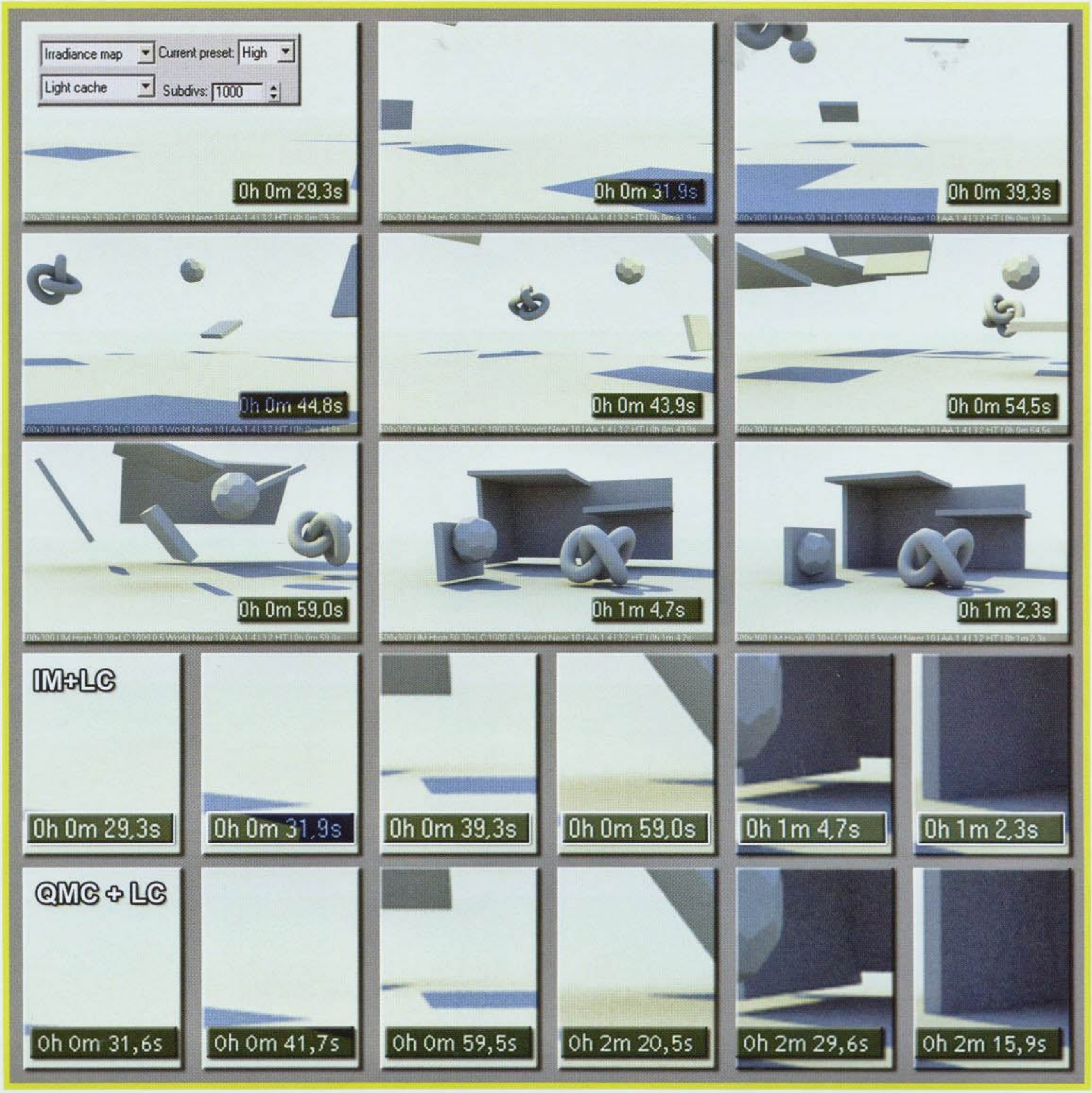


■ **Figure 3.261**
Rendering of a dynamic outdoor scene via the ***QMC+LC*** method.

In order to render the test scene, the ***QMC*** method has been used as ***Subdivision*** = 20. It is a medium value which for this scene is enough to obtain a clean render. ***LC*** is set to single frame mode, as GI has been calculated for every single frame. This way rendering times have been obtained from 30 sec. up to 2 min 29 sec. Noise is present in the areas with indirect light.

IM + LC

In this particular type of scene, the two systems for GI calculation must be set to ***Single frame*** mode. Still, with this pair flickering problems may occur, mainly due to low values in the settings. In order to solve this phenomenon, one must increase the GI calculation parameters for both ***IM*** and ***LC***.
In the next example, the first frames of the animation are very simple. Thus rendering time is also very low. As it proceeds, the animation becomes ever more complex. New geometrical objects take their place and rendering time starts to increase, although remaining lower than the ones encountered with the ***QMC+LC*** method. Noise is obviously inexistent because of the nature of ***IM*** and ***LC***. Having used high parameters there is no flickering.



■ **Figure 3.262**
Comparison between the ***QMC+LC*** method and the ***IM+LC*** in a dynamic outdoor animation.

CHAPTER 4: RENDERER - PART 2

VRAY: RENDERER - PART 2

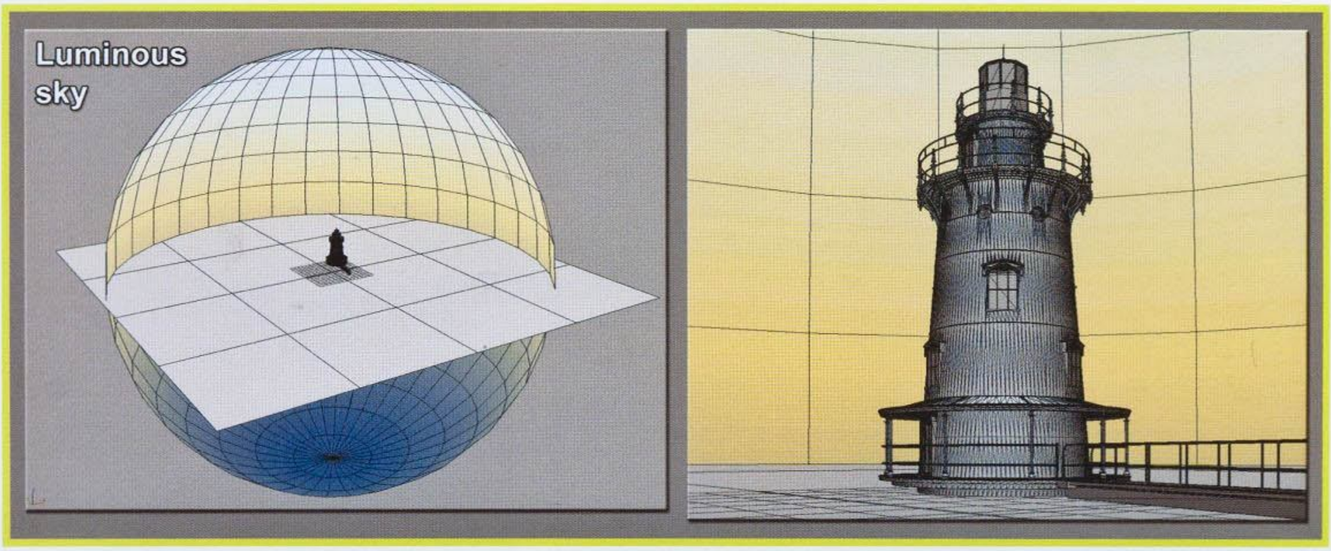
In this section the remaining sections of the wide panel Renderer of *VRay* are to be analyzed. Here below, in order, the rollouts which will be described:

- VRay: Environment.
- VRay: Caustics.
- VRay: rQMC sampler.
- VRay: Color Mapping.
- VRay: Camera.
- VRay: Default displacement.
- VRay: System.

VRay: Environment

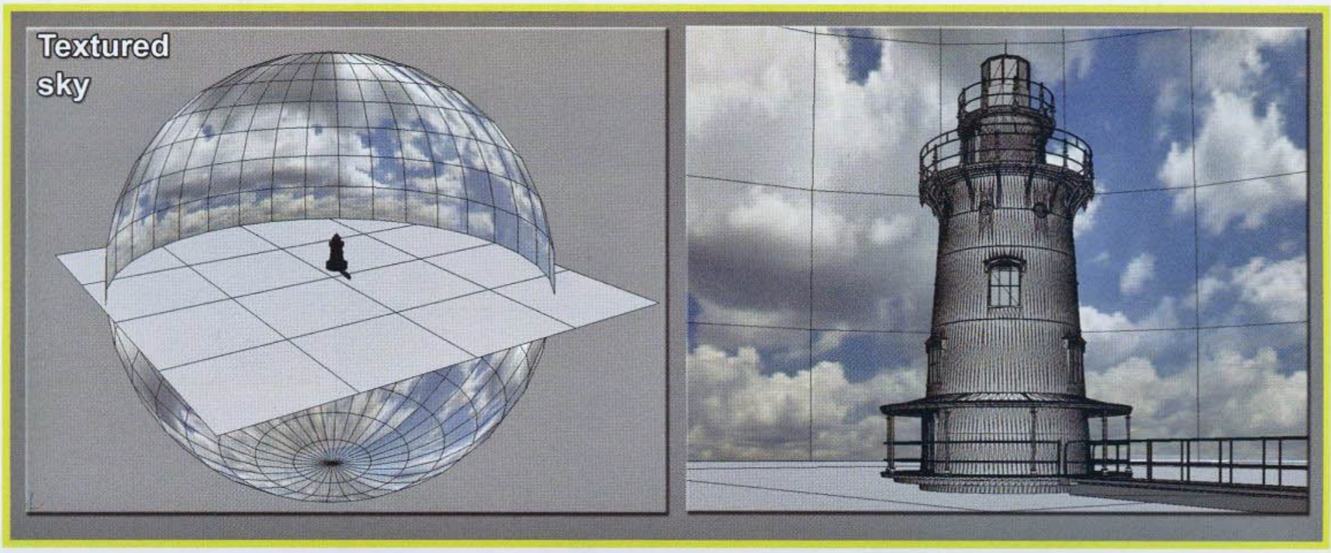
INTRODUCTION

When creating a new file within 3ds Max, three grids are usually shown to the user in the work window. Each grid represents the plane for the axis X, Y, and Z of the three-dimensional space where the scene is to be modelled. This space does not have limits in any direction. In any case, it is possible to define the boundaries of the scene, by manually creating a space whose edges are geometrically precisely defined.
The aim is to create a model inside a discrete space, outlined in every direction. Imagine introducing a fake sky, a sphere, with very big radius. A three-dimensional model is to be inserted in it. To be realistic, it is then possible to apply to the sphere a map, which can be either textured with colors or with textures. This geometry, which has the usual inverted faces to be visible from its insides, is able to produce light through the applied colour or the map, and generate the Global Illumination. This way, the model positioned in the centre of the sphere is to be lightened in each point of the sphere surrounding it by 360°.



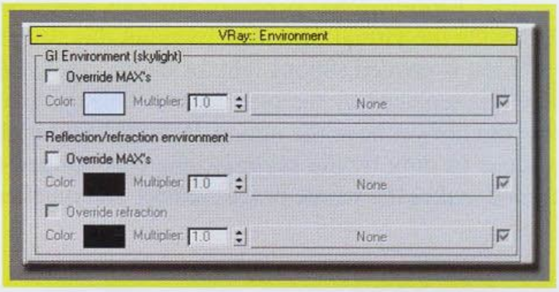
■ **Figure 4.1**
Spherical geometry representing the sky. This geometry has self-lightening properties.

Beyond lighting the scene, it is possible to give this huge sphere the task of creating reflections. For example, if one wants to render a chrome mesh, it is possible to make this object have reflections even though there is no other object to be reflected around it. If a map is used for texturing the sphere, then it is possible to reflect it in the mesh itself. In case a texture of the sky is used, the object has reflections of the surrounding environment, that is the mapped sphere.



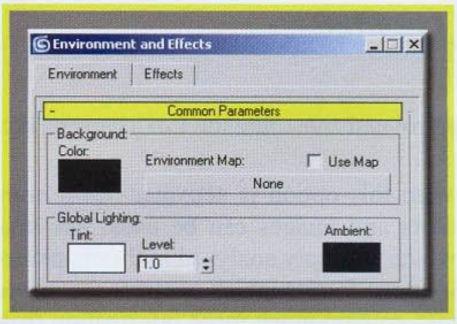
■ **Figure 4.2**
Use of a texture for mapping the Environment.

With *VRay* the geometrical creation of a sky, a sphere is not necessary. What has been said up to now is to be used for the comprehension of how the ***Environment*** of *VRay* works. In fact, in the panel of *VRay*, only some selectors of colors and some buttons disabled by default appear. These are simply functions allowing one to apply color or texture to the large geometry previously described. When the GI is activated, *VRay* virtually creates this huge sphere, even if it is not physically visible neither on the scene, neither in the list of objects, nor in the viewport. *VRay* always considers it, and the user only has the task of setting the color, the texture, the intensity to give to the generated light, etc…



■ **Figure 4.3**
Environment panel in VRay.

When the GI is activated and no light is present on the scene, *VRay* uses the light generated by the Environment of 3ds Max as an environment light. Why 3ds Max? Because *VRay* has by default the option ***Override MAX's*** disabled, option which is to be found in the ***Environment*** of *VRay*. This means that *VRay* uses the color or the texture present in the Environment panel of 3ds Max for generating the GI, reflections and refractions.



■ **Figure 4.4**
Environment panel in 3ds Max.

The benefit of this peculiarity, which might appear strange, is soon explained. In case one wants to use an image of the sky as a background of the scene, the obvious solution would be to insert it in the button Environment Map of 3ds Max's rollout Common Parameters. Nothing strange up to now.

But at the moment of the activation of the GI, *VRay* would use the image which just been loaded as a light source. Sometimes this is not what the user wants. By activating the parameter ***Override MAX's,*** *VRay*'s ***Environment*** is overwritten on 3ds Max's one, leaving in this way as background image the sky and delegating to the *VRay* 's ***Environment*** the computation of the GI.
In this example a green ***Environment*** has been used to emphasize the difference between the sky, simple spherical map applied in 3ds Max's Environment, and *VRay*'s one. The color of the material of all objects on the scene is a medium tone of grey.

■ **Figure 4.5**
Render with background managed by the Environment of 3ds Max and Global Illumination generated by the settings of VRay's ***Environment***.

The green diffuse illumination created by the ***Environment*** of *VRay* involves the entire scene, while the background is managed by the Environment of 3ds Max.

It is necessary to pay attention to another thing: remember the deactivation of the 3ds Max default light! As explained in the chapter devoted to the rollout ***VRay: Global switches***, 3ds Max lightens the scene with a general default light, which allows one to see the model inside the virtual 3D space (without this illumination the space and the viewport would be completely dark). When any light is created (Spot Light, or Omni Light, ***VRayLight***, etc...) the default light of 3ds Max is automatically disabled and it is replaced by the present one. This used to be a trick for the deactivation of the default light of 3ds Max when *VRay* did not yet have the option ***Default Light*** in the rollout ***VRay: Global switches***. In fact, in order to correctly render a scene lit only by the ***skylight***, one was obliged to create a light and then deactivate it. Now this useful parameter exists, and makes the whole process easier and quicker.

PARAMETERS

GI Environment (skylight)

All of these parameters control and substitute the Environment function of 3ds Max.

Override MAX's - when this option is activated *VRay* uses its ***Environment*** or ***skylight*** for generating the GI inside the scene. Both colors and images can be used. For images a specific texture is used, named ***HDRI*** (**H**igh **D**ynamic **R**ange **I**mage), which is deeply analyzed in the chapter concerning textures. For the moment it can be said that the ***HDRIs*** are a particular image format which contains the whole range of luminous intensities, from the lowest to the highest. That is these images contain much more information than a common RGB image. This additional data is used to "produce" light used by *VRay* for GI computation.

Color - it specifies the color of the ***Environment*** (***skylight***). All 16.7 million colors of the chromatic range can be used.

■ **Figure 4.6**
Renderer using the only ***skylight*** color with the method ***QMC+QMC***. 500x500 pixels rendering dimensions.

In the test above, each geometry has a neutral color, a light grey in RGB = [190, 190, 190]. The uniform color light is instead given by ***skylight***'s setting. The brighter the ***Environment*** color is, the higher the rendering time is. This happens because the light is more intense and has a higher energy. Hence, its distribution requires more time, because more bounces are involved. Furthermore, with the color it is possible to define the intensity of the ***skylight***. The darker the color, the lower the luminosity of the scene.

Multiplier - it specifies the multiplier and the intensity of color. This parameter does not affect the texture beneath the spinner of the ***Multiplier***.

■ **Figure 4.7**
Change of ***skylight*** intensity of VRay by using the ***Multiplier*** spinner.

Textures - it allows one to specify a texture or a map to be used as ***skylight***. In 99% of cases ***HDRI*** maps are used, as they simulate real light perfectly. There are many companies producing interesting collections in high resolution. Websites offering them freely also exist. They might also be "home-created", simply by using compact digital cameras.

■ **Figure 4.8**
VRay allows one to use particular maps for the generation of the GI thanks to the ***Environment***, beyond the simple color.

In the test two kinds of maps have been used allowing the generation of the GI thanks to the ***Environment***: the ***HDRI*** and the ***VRaySky***. These textures are to be more deeply analyzed further on, but it is already possible to make some considerations. ***HDRI*** maps have a "strange" shape. They are maps which have an unusual conformation. In this case so called spherical maps have been used. Thanks to this peculiarity they fit perfectly for being applied to the fake sphere without giving any problems. ***VRaySky*** is also a spherical map, but in this case it is parametric, meaning that it is mathematically generated by ***VRay***'s algorithms. In this case its roundness is less visible, as it is composed by color gradients.

It can be noticed how the color of the ***HDRI*** involves the model realistically, preserving the position of the main light source too, providing that it exists in the texture. In fact, in the render with the ***HDRI*** of the sky it can be noticed how the sun, the main light source, is placed on the left. In this way it is possible to see how the lighthouse is lit mostly on the left side. The same goes for the internal ***HDRI***, with interesting ivory-colored fading caused by the walls of the room. With ***VRaySky*** this phenomenon is not present, due to its flatness, and the scene is uniformly illuminated from all directions.

As far as rendering times are concerned an observation can be made. The render of the room carried out with the ***HDRI*** is twice as slow as the external one. This is caused by the ***HDRI*** map and its quality. The ***HDRI*** of the room is very detailed, much more than the other maps used, but most of all it has a higher resolution. This causes more calculations for ***VRay*** compared to the one with the sky, which is flatter and without details.

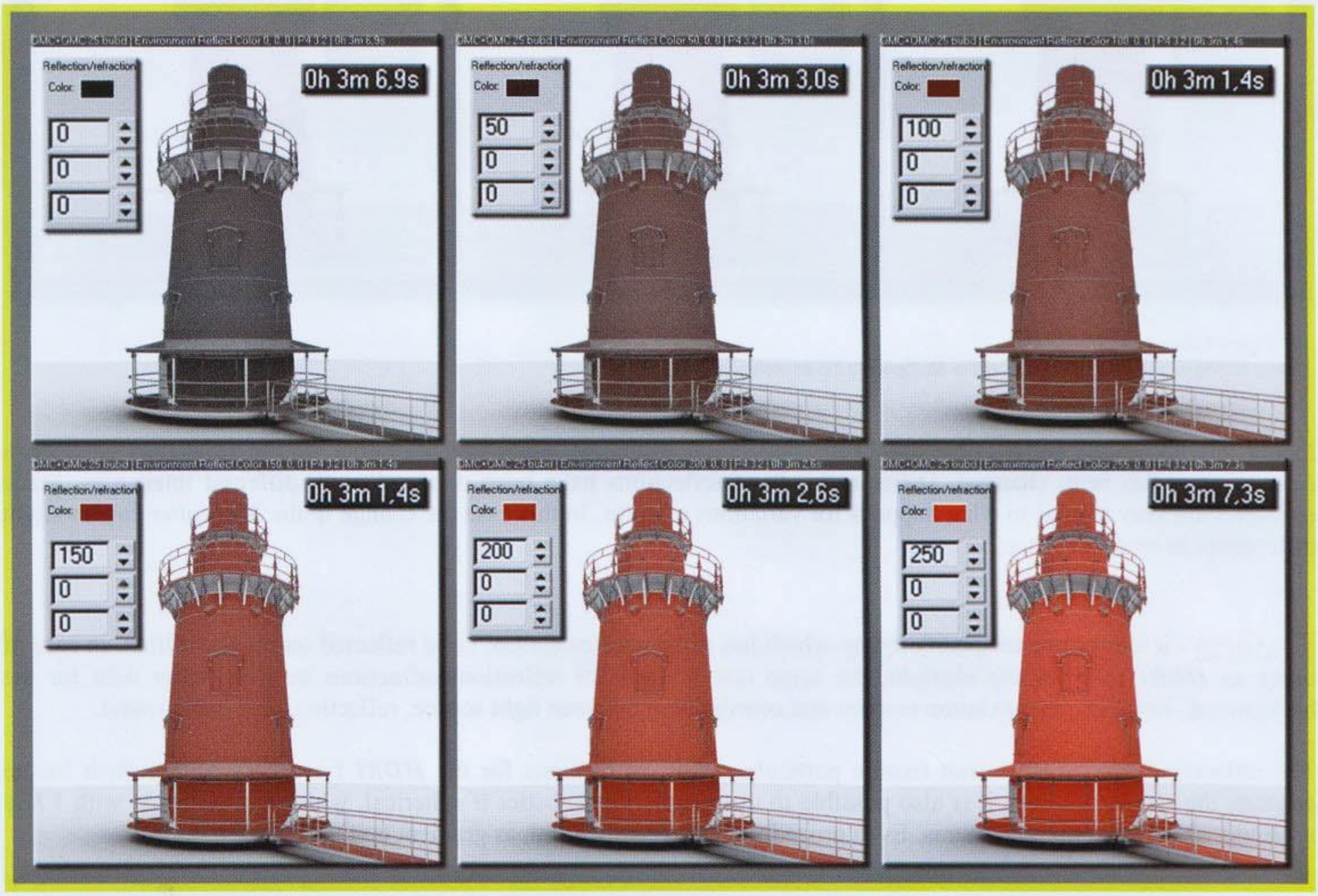
In conclusion, it can be noticed how the same map ***HDRI*** has been used both for the generation of the GI and for the background. But it is also possible to use different maps, one for the generation of indirect illumination (usually an ***HDRI*** or the ***VRaySky***), and one for the background, loaded in the ***Environment*** of 3ds Max. This texture can be any one among the maps supported by 3ds Max, ***HDRI***, Bitmap and procedural ones too.

Reflection/refraction Environment

As with the ***skylight***, it is possible to use separate maps for controlling both global reflections and refractions. This last subdivision is a new feature introduced in version 1.5. Reflections and refractions have the same parameters. Thus only the ones for the reflections have been analyzed. For refractions the explanation of the parameters is the same, just applied to different physical facets.

Override MAX's - with this option one has the possibility of using, for the reflections, specific colors and maps managed by *VRay*, instead of the Environment of 3ds Max. This way the user is free from 3ds Max and can set a color or a map independently of the background.

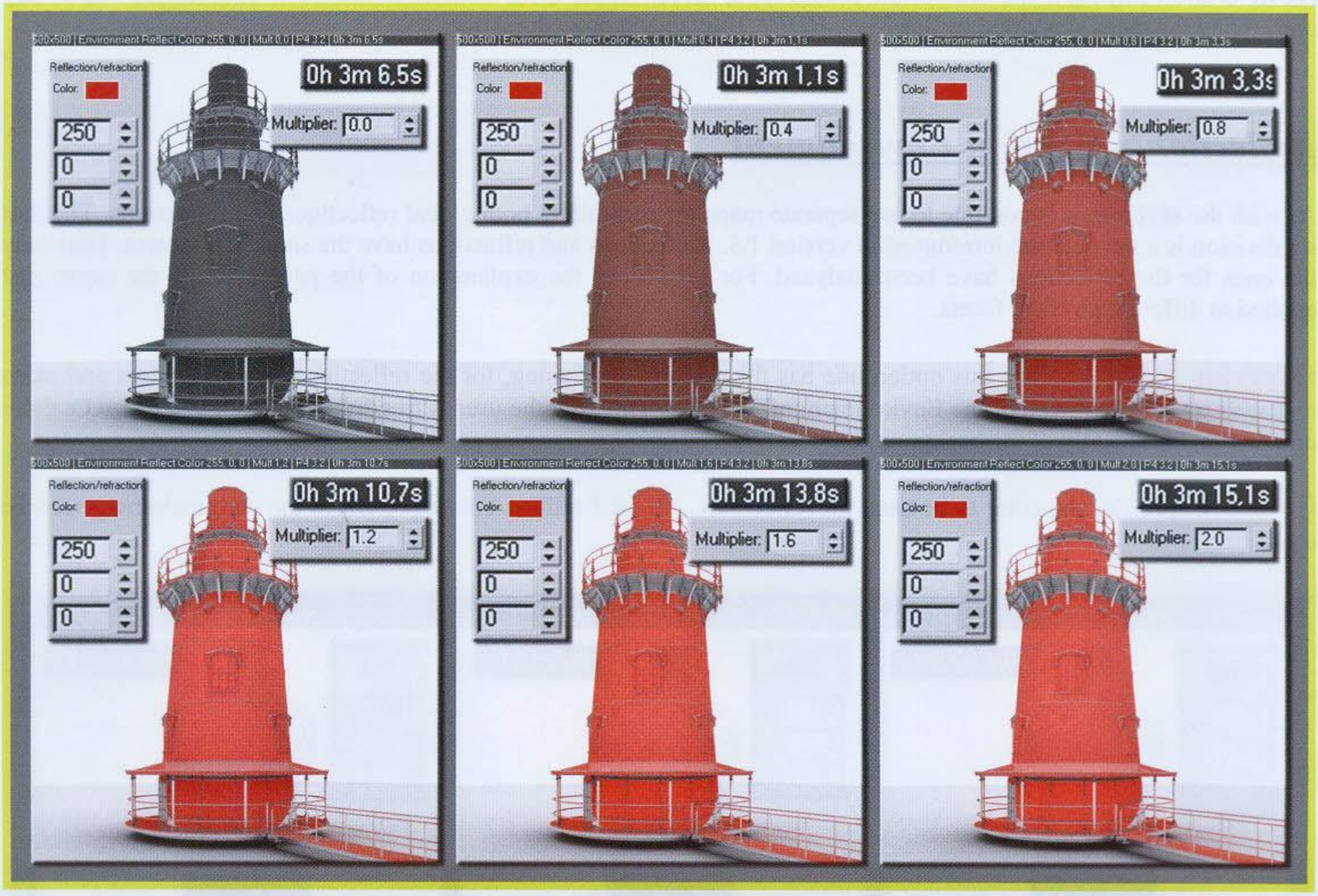
Color - it specifies the color to be used for reflections. All 16.7 million colors present in the chromatic range can be used.



■ **Figure 4.9**
Render with different ***Color*** values for reflections, found in VRay's ***Environment***.

In the test above the lighthouse has a uniform grey color RGB = [190, 190, 190], with a medium-high reflection RGB=[150,150,150]. The plane is grey as well, without any reflection applied. As background and ***skylight*** a very bright grey has been used, nearly white. For reflections several hues of red have been used, from the dark one to the brighter one. This directly affects the reflection on the object. A dark reflection-color of the ***Environment*** generates a dark reflection. Bright colors cause brilliant reflections. In this case, changing the reflection-color, there is no change in the rendering time.

Multiplier - it specifies the multiplier of the color. It has to be noticed that this parameter does not affect at all the texture just beneath the spinner itself.



■ **Figure 4.10**
Tests carried out with different values of the ***Multiplier*** of the environmental reflection.

In the test above the same color has been used in all examples. Only the multiplier of the reflection of *VRay*'s ***Environment*** has been changed. Different kinds of reflections have been obtained, with different intensities. Some examples are very similar to what happens for variations of color. In this case the change in the ***Multiplier*** causes slight differences in rendering time.

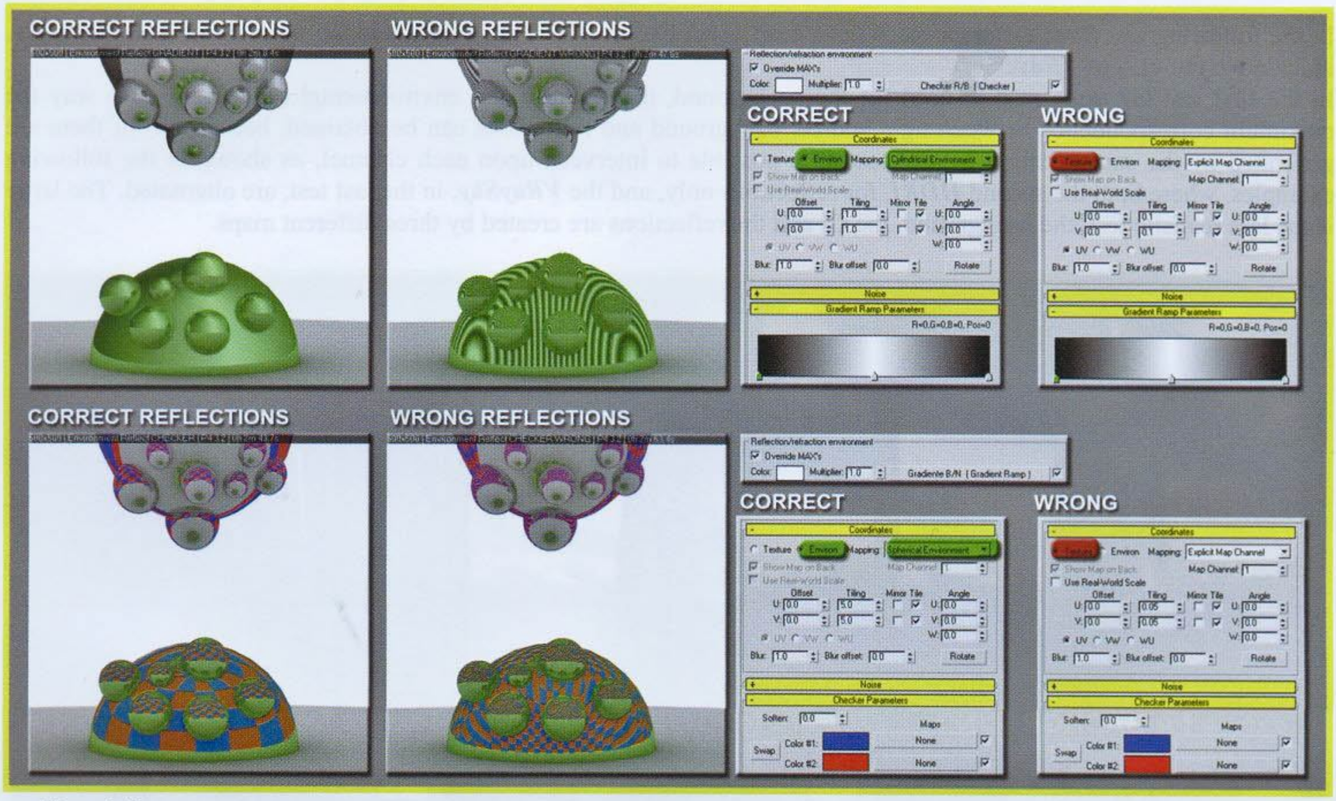
Textures - it allows one to specify a map which has to be used as texture to be reflected on objects. Often, in case of using an ***HDRI*** map for the ***skylight***, the same one is used for reflections/refractions as well, other than for the background. This assures maximum realism and coordination between light source, reflections and background.

The reflections/refractions do not require particular maps, as happens for the ***HDRI*** for the ***skylight***, which has to generate the GI. In this case it is also possible to use simple maps, better if spherical, which can be made with *VRay* itself too, using appropriate cameras. Even procedural maps can be used, as gradient textures.

In the following test three different maps have been used, two of which are ***HDRI*** and the other is the new ***VRaySky***, included in the v1.5 of *VRay*.
In the first test the same map is used for the background, the ***skylight*** and environmental reflections. This way the maximum correspondence between light source, background and reflections can be obtained, because all of them are generated by the same texture. In any case it is possible to intervene upon each channel, as shown in the following examples, where both the second ***HDRI***, for reflections only, and the ***VRaySky***, in the last test, are alternated. The latter is the less realistic test: the background, the GI and the reflections are created by three different maps.

■ **Figure 4.11**
Use of different maps for the ***Background***, generation of the GI through the ***Environment*** of VRay and environmental reflections.

It is then possible to use standard maps, such as the Gradient Ramp, Checker, etc... to be used as texture for reflections/refractions. The only thing to remember, in this case, is to modify the mapping coordinates in Environment, using the Spherical or Cylindrical for mapping.
In the example these two maps have been used, set up in such a way to be projected in a spherical and cylindrical shape in the Environment of 3ds Max. This way the reflection appears correct. Spheres fit very well in illustrating this phenomenon. It can be noticed how reflections are correct only when used with Environ coordinates.

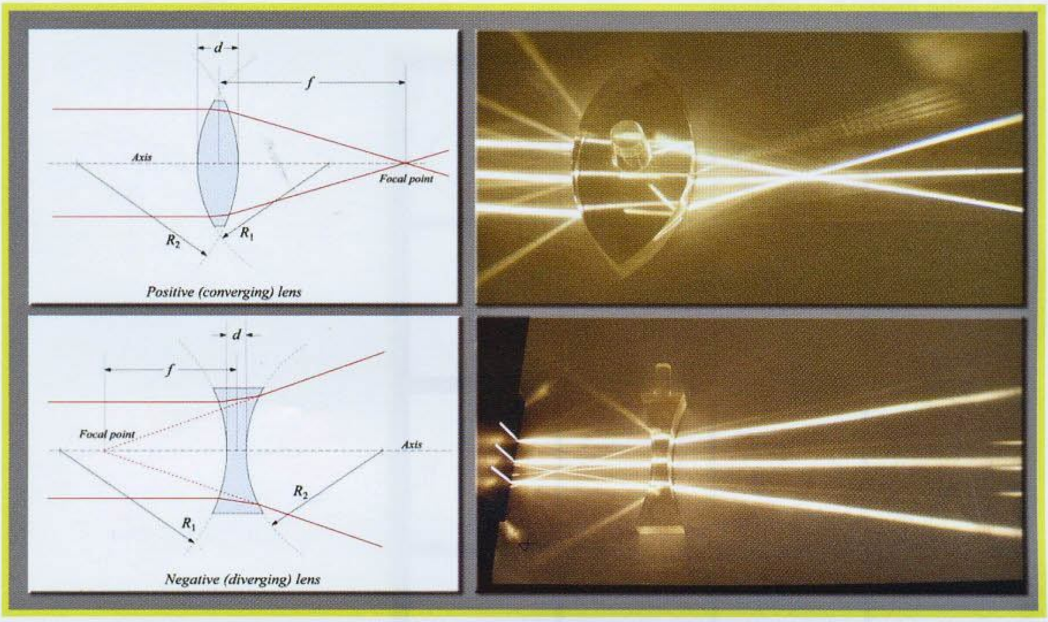


■ **Figure 4.12**
Setting of environmental reflections using simple procedural maps in 3ds Max.

VRay: Caustics

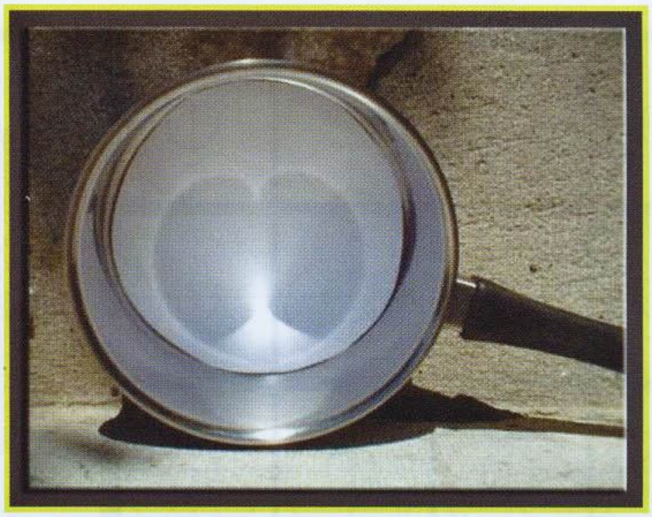
INTRODUCTION

Objects with concave conical sections have the property of converging light that passes through them in one spot. On the contrary, if sections are convex, rays diverge. This phenomenon can be observed in optical lenses.



■ **Figure 4.13**
Images showing the typical behaviour of light when it passes through both concave and convex lenses, generating refraction caustics.

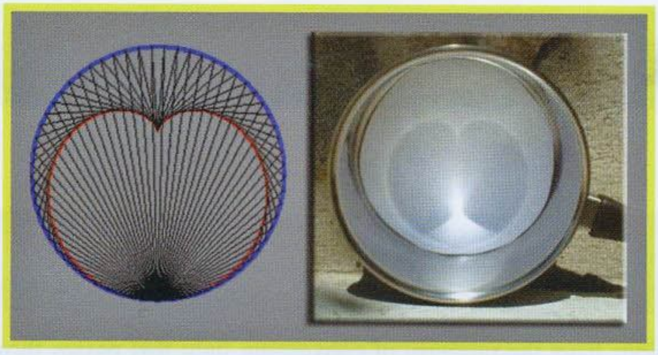
Caustics are phenomena forming in nature which can be observed in any moment during the day. They can be generated by refraction, as seen in the case of the lenses, where rays diverge because they pass through a medium with different density, or by reflection. One might carry out an experiment with a pan, obliquely lit. In this case rays reflect on the vertical surface, creating on the bottom of the pan a curve, called reflection caustic, because it is obtained by the reflection of luminous rays.



■ **Figure 4.14**
Caustic effect by reflection.

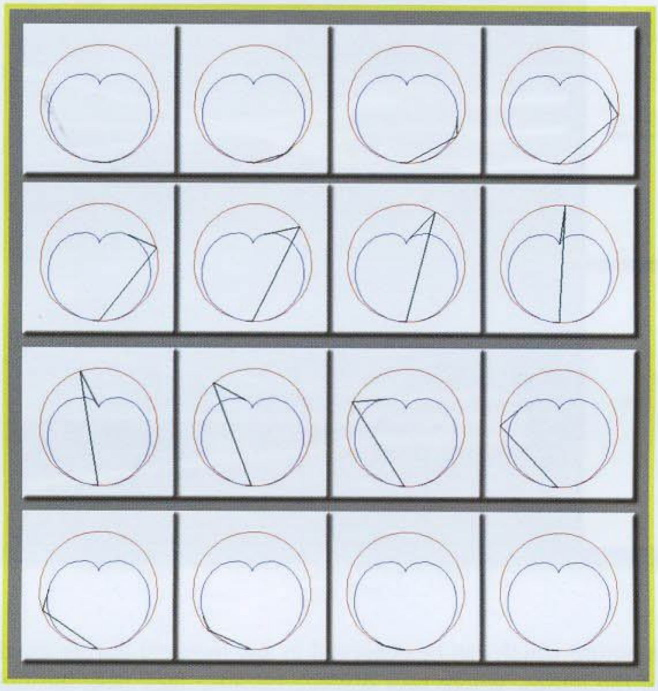
This particular phenomenon happens because rays are reflected by an object strictly following physical laws. The classic example is the geometrical shape of a cardioid. The cardioid is a mathematical curve which can be described in different ways. One of these is precisely the use of caustics by reflection.

■ **Figure 4.15**
Comparison between a drawn cardioid and a real caustic.

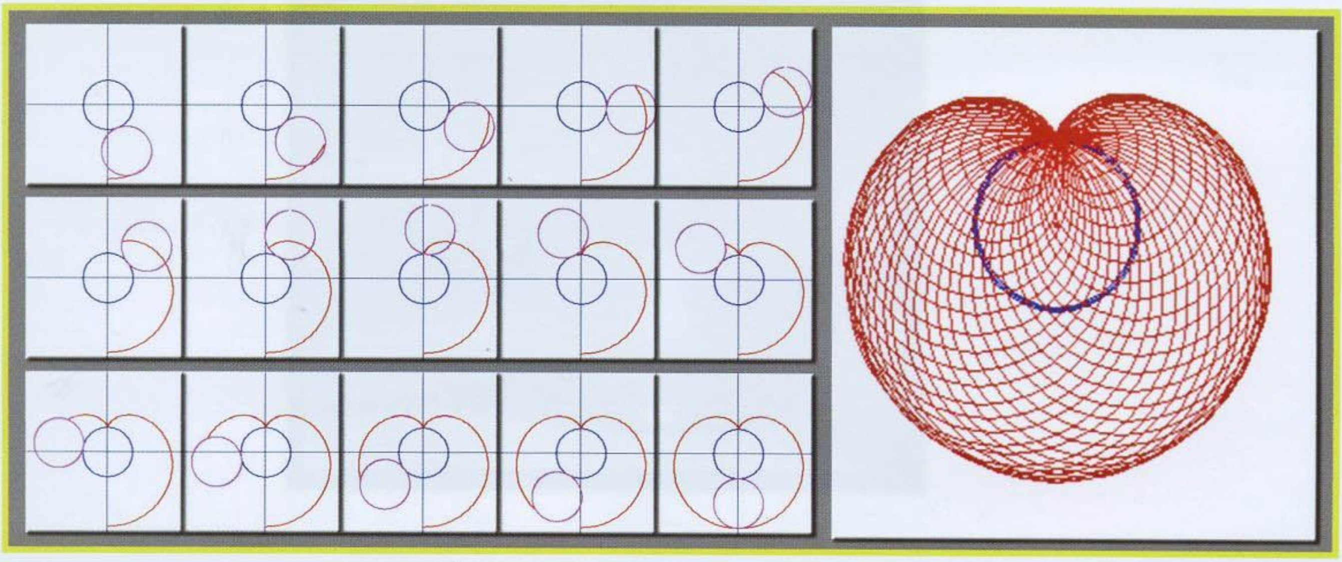


In order to create a cardioid by reflection one has to draw some lines, starting from a point on the circumference, directed inwards. After that, one traces the normal in the intersection point, between the circumference and the radius which has been drawn. The resulting image is a cardioid.

■ **Figure 4.16**
Creation of a cardioid by reflection.

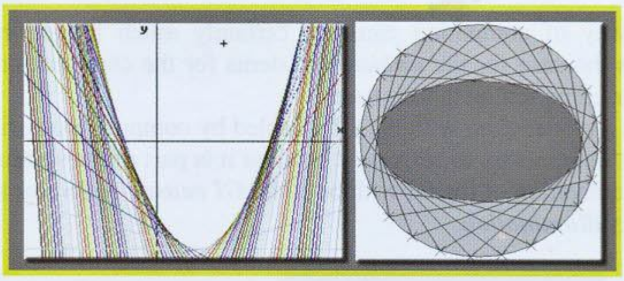


By way of information, many other methods for creating cardioids exist. One of these is, for example, a fixed point (red) in a circumference (pink) which is rolling around another motionless circumference (blue) with the same radius.



■ **Figure 4.17**
Cardioid obtained by the rolling of a circle.

A reflection caustic can also be defined by the envelope of reflected rays.

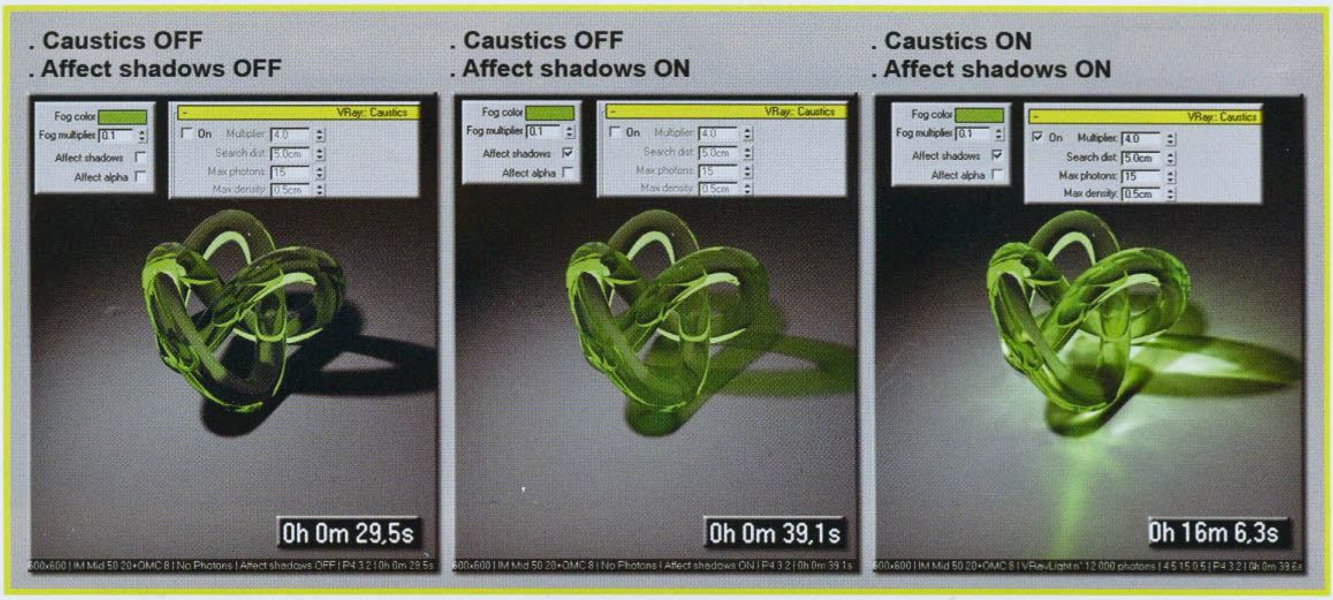


■ **Figure 4.18**
Representation of geometric figures (parabola and ellipse) through an envelope of lines.

How are caustics produced inside of *VRay*? There are two systems for the computation of the caustics: an approximate one and a non-approximate one.

APPROXIMATE METHOD

It is the classic method and the most used one, producing excellent results in little time. The generation of caustics in this case is produced by **photons**. As for the ***GI Photon Map***, subject which has already discussed, in this case too it is fundamental to have a light source able to emit photons. Self-lighting objects and ***skylight*** are not able to produce caustics. Another very important condition is to have objects within the scene with reflective or refractive properties.
When we later discuss *VRay*'s shaders, the parameter ***Affect shadows*** will be described. It is a very important function for caustics. By activating it, *VRay* "cheats" on the idea of "physically correct" and allows light to pass through a semi-transparent object by means of a "fake". When caustics are activated through photons, ***Affect shadows*** is no longer considered by *VRay*, which considers it disabled even though it is not, delegating to caustics the correct behaviour of the light through a refractive object. In this way the "physically correct" factor is restored.



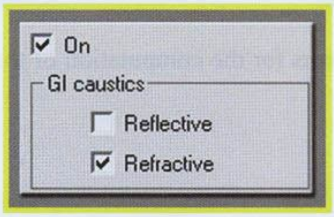
■ **Figure 4.19**
Creation of caustics thanks to photons. The scene is composed by a ***VRayLight***, a plane and a geometry. The GI is active, with the ***skylight*** reset to zero.

In the first example both the caustics and the option ***Affect shadows*** are disabled. Light does not generate any effect and neither does it pass through any geometry. In the second example ***Affect shadows*** is active and in this case the shadow of the geometry is colored because of the refraction of the light. In the last example caustics produced by photons have been activated. Even though the option ***Affect shadows*** is active, it is not considered: only caustics determine the correct behaviour of light and shadows.
This method is called approximate because photons are used to compute caustics. Much like what happens for the GI, with photons it is possible to have punctual information about caustics. These values have then to be interpolated, in order to meet the lack of information due to the use of a discrete number of photons. The higher the number of photons used in the computation, the higher the quality of caustics is. All this is very similar to what happens in the ***Photon Map*** used for the Global Illumination.

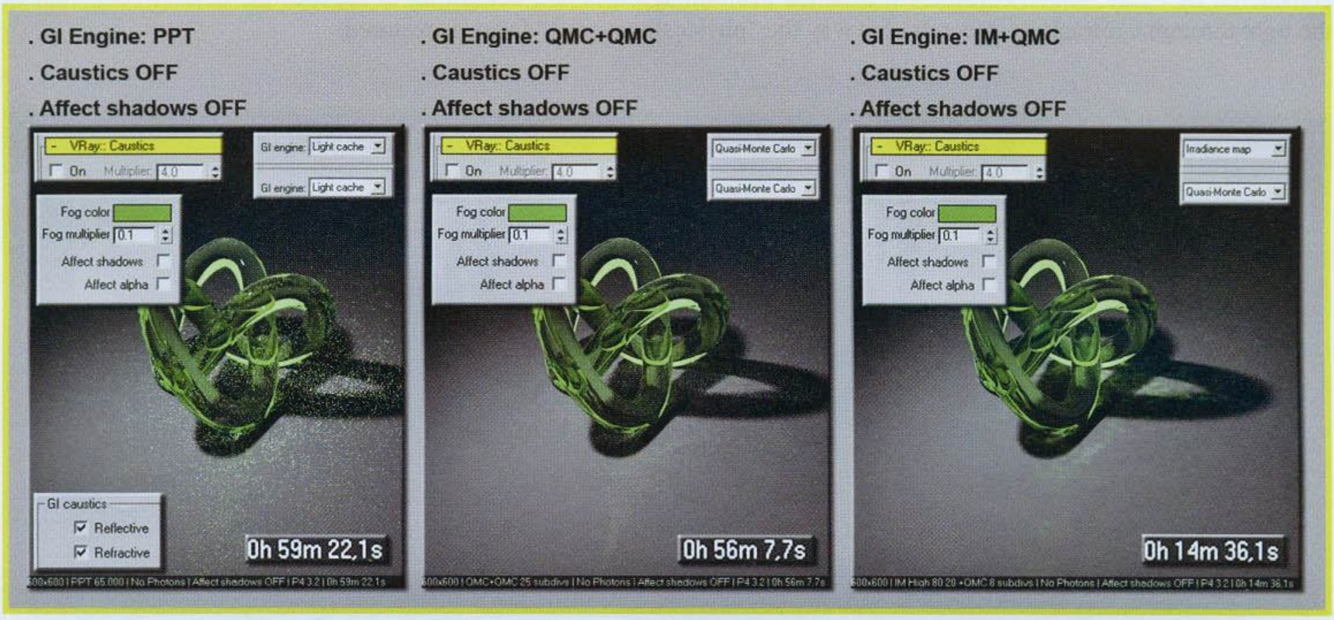
NON-APPROXIMATE METHOD

VRay enables another way of computing caustics, certainly much longer, but very accurate and with very few parameters to set, as was the case for the ***unbiased*** systems for the computation of the GI. With the non-approximate method no particular approximation method is used.
It is possible to carry out a render even with the GI disabled by computing caustics with photons. Instead, with the non-approximate method, it is compulsory to activate it because it is part of the system for the generation of caustics. In fact, the only operation is the activation of the GI with both the ***GI caustic reflective*** and ***GI caustic refractive*** parameters in the rollout ***VRay: Indirect illumination***.

■ **Figure 4.20**
Default parameters for the management of caustics created by the Global Illumination.



In this case too it is necessary to pay attention to the ***Affect shadows*** value. By activating it, *VRay* forbids any kind of phenomenon of caustics, because it tries to cheat the physicality of the result, creating a "fake" for light behaviour, as happens in the approximate method. By disabling it, instead, caustics are computed by the GI engine in a physically correct way, without fakes.
When using the non-approximate method it is necessary to consider the light sources too. When using photons for the generation of caustics it is necessary to use sources able to emit photons, such as Direct Light, Spot Light or ***VRayLight***. For non-approximate systems usually ***VRayLight*** or self-lighting objects are usable, because the lights of 3ds Max do not produce any caustic effect with the non-approximate system.



■ **Figure 4.21**
Computation of caustics using the GI only.

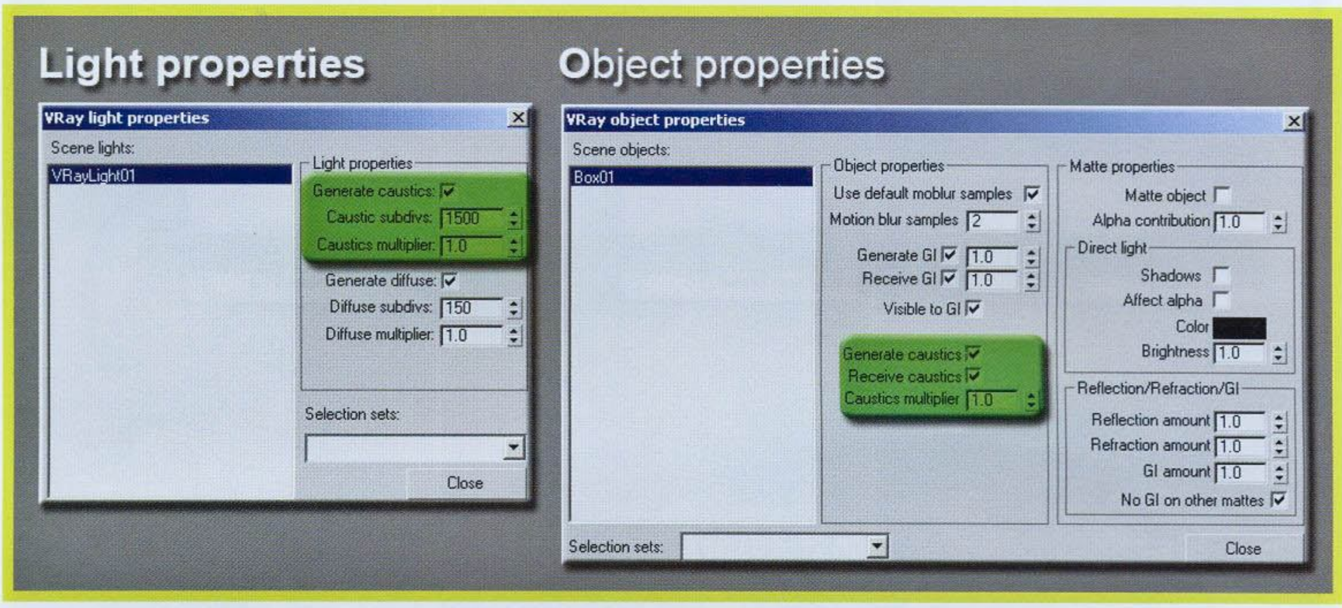
In these tests the same scene used for the approximate case has been used. The only difference is the deactivation of photons. The GI has the task of computing caustics this time.
In the fist case the ***PPT*** has been used. There is nothing to set, except activating the GI. One hour after the beginning of rendering the presence of caustics is evident, but not very homogeneous and totally unsatisfactory. The same goes for the method ***QMC+QMC***. With the number of ***Subdivs*** = 25, in an hour, the quality of caustics, although recognizable, is very low. The use of the ***IM*** is not recommended. As can be noticed in the test, caustics appear more as stains, "blotches", which might seem more like mistakes rather than real caustics in the rendering. Thus, it is advisable to always activate the option ***Affect shadows*** together with the use of the ***IM***, to avoid these kind of problems. Another phenomenon is also visible. With the ***Photon Map*** it is possible to manage the intensity and brightness of caustics. This is not possible with the non-approximate method. All computation is in fact managed by *VRay*, which automatically sets the right intensity in order to have a physically correct result.
Therefore the explanation of the approximate method alone follows.

PARAMETERS

For the generation of caustics two things are essential:

. a light source able to emit photons.
. reflecting or refracting objects, generators of caustics.

Inside *VRay* there are many parameters which allow the management of caustics. Some of these parameters can be modified both globally, inside the rollout ***VRay: Caustics***, and locally. The local parameters regard two classes: ***objects*** and the light sources (***light***). These values are positioned in the ***properties*** window of *VRay*. In these panels, together with other parameters which are to be discussed further on, also the ones controlling the caustics are present.



■ **Figure 4.22**
Panels of objects and lights properties of VRay. Inside them some parameters for the management of the caustics are included.

VRay LIGHT PROPERTIES

Generate caustics - if active, the light produces photons for caustics computation. With the light disabled, the light source is not considered in the computation, but it continues to light the scene, it generates the GI and the shadows. The deactivation or activation of this parameter makes a new computation of the caustics necessary.

■ **Figure 4.23**
Scene rendered with the ***Generate caustics*** parameter deactivated and activated.

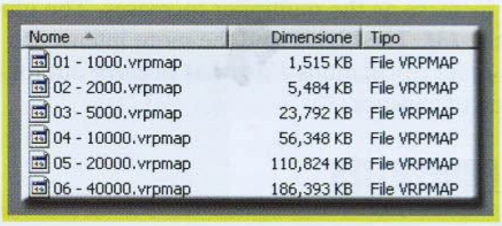
This scene, made up of five geometries (two glasses, a ring, the flooring and the vertical wall), is illuminated by a ***VRayLight***, in addition to the ***skylight***. In both tests caustics have been activated. In the left-hand one the option for the ***VRayLight Generate Caustics*** is disabled. Vice versa, when the option ***Generate caustics*** is activated, caustics become visible, as the ***VRayLight*** is able to emit photons for generating them.

Caustics subdivs - this option controls the amount of photons emitted by the light source, which *VRay* uses for the computation of caustics. High levels mean a higher number of photons emitted, a better quality of the caustics, but a higher consumption of memory and greater rendering time too.

■ **Figure 4.24**
Change of the ***Subdivs*** and the ***VRayLight***. This variation requires a new computation of caustics.

The test speaks clear. Increasing the number of ***Caustics subdivs*** the quality of the render increases as well. Studying it well it can be noticed how the quality of caustics, having reached a certain threshold of ***Caustics subdivs***, increases very slowly if compared to rendering time. Already with 20,000 ***Caustics subdivs*** the quality is very high and to obtain a greater quality it is necessary to reach a rendering time of 1h and 30 min.
It can be noticed how the time taken for the computation of caustics multiplies by four when subdivisions double and this happens for the size of the map too. This occurs because the value ***Caustics subdivs*** does not represent the number of photons used, a value obtained instead by squaring it.

1,000	Subdivs	=	1,000,000	photons
2,000	Subdivs	=	4,000,000	photons
5,000	Subdivs	=	25,000,000	photons
10,000	Subdivs	=	100,000,000	photons
20,000	Subdivs	=	400,000,000	photons
40,000	Subdivs	=	1,200,000,000	photons



Caustics multiplier - this option locally controls the intensity of caustics for each single light. This value of multiplier is cumulative. In fact, another multiplier at a global level exists, which allows the management of all the light sources present in the scene. As a consequence:

Local multiplier		Global multiplier		Effective multiplier
1	x	1	=	1
2	x	1	=	2
2	x	2	=	4

■ **Figure 4.25**
For physically correct renderings it is advisable not to use values higher than 1.

Rendering times increase as the multiplier increases. The glow generated by caustics becomes part of the computation of the GI inside the *IM*. The higher and the more intense the light in the scene is, the higher the number of bounces is, and therefore the more calculations *VRay* has to carry out for computing the GI. The variation of this parameter requires a new computation of caustics.

VRay OBJECT PROPERTIES

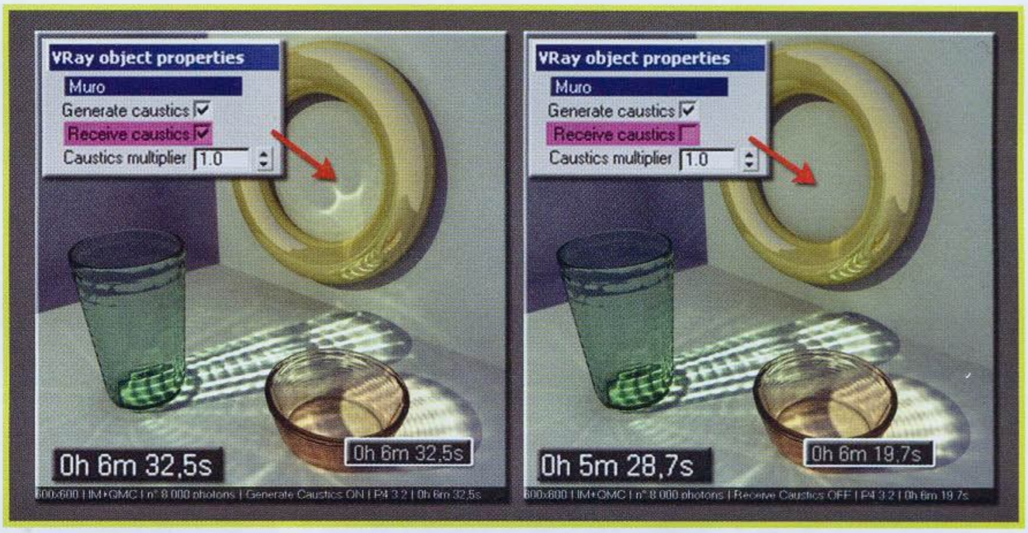
Generate caustics - if activated for objects with reflection or refraction properties, this parameter allows the generation of caustics for the selected object. This way it is possible to define accurately which geometries on the scene are able to produce caustics.

■ Figure 4.26
Example of activation and deactivation of the property for object ***Generate caustics***.

In this example the green glass has been excluded from the computation of caustics. In addition to the reduction of the rendering time due to the lesser amount of caustics to be computed, the glass does not have a realistic shadow. When the option ***Generate caustics*** is activated or deactivated a new computation of the caustics is necessary.

Receive caustics - if activated, this parameter allows the selected object to receive caustics.

■ Figure 4.27
Example of activation and deactivation of the property for objects ***Receive caustics***.

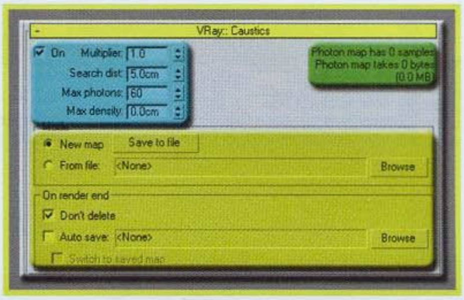


The wall, which has the possibility of receiving caustics from other objects in the first test, in the second one does not have this possibility anymore, and the function ***Receive caustics*** has been deactivated. This way caustics produced both from the glasses and the ring are not present. A new computation is necessary when the option ***Receive caustics*** is activated or deactivated again.

Caustic multiplier - this multiplier allows one to increase or decrease the intensity due to caustics produced by the selected object.

Rollout VRay: Caustics

This rollout is contained in the main system of *VRay*, inside the Renderer panel. It contains fundamental parameters for the control and management of the ***Photon Map***, and it is subdivided in three parts. The first one, probably the most important one, in where four spinners are to be found, allows one to set the quality of the caustics. The second one contains settings for creating, loading and saving the generated map. The third one, not editable by the user, gives some information concerning the computation of the ***Photon Map***.

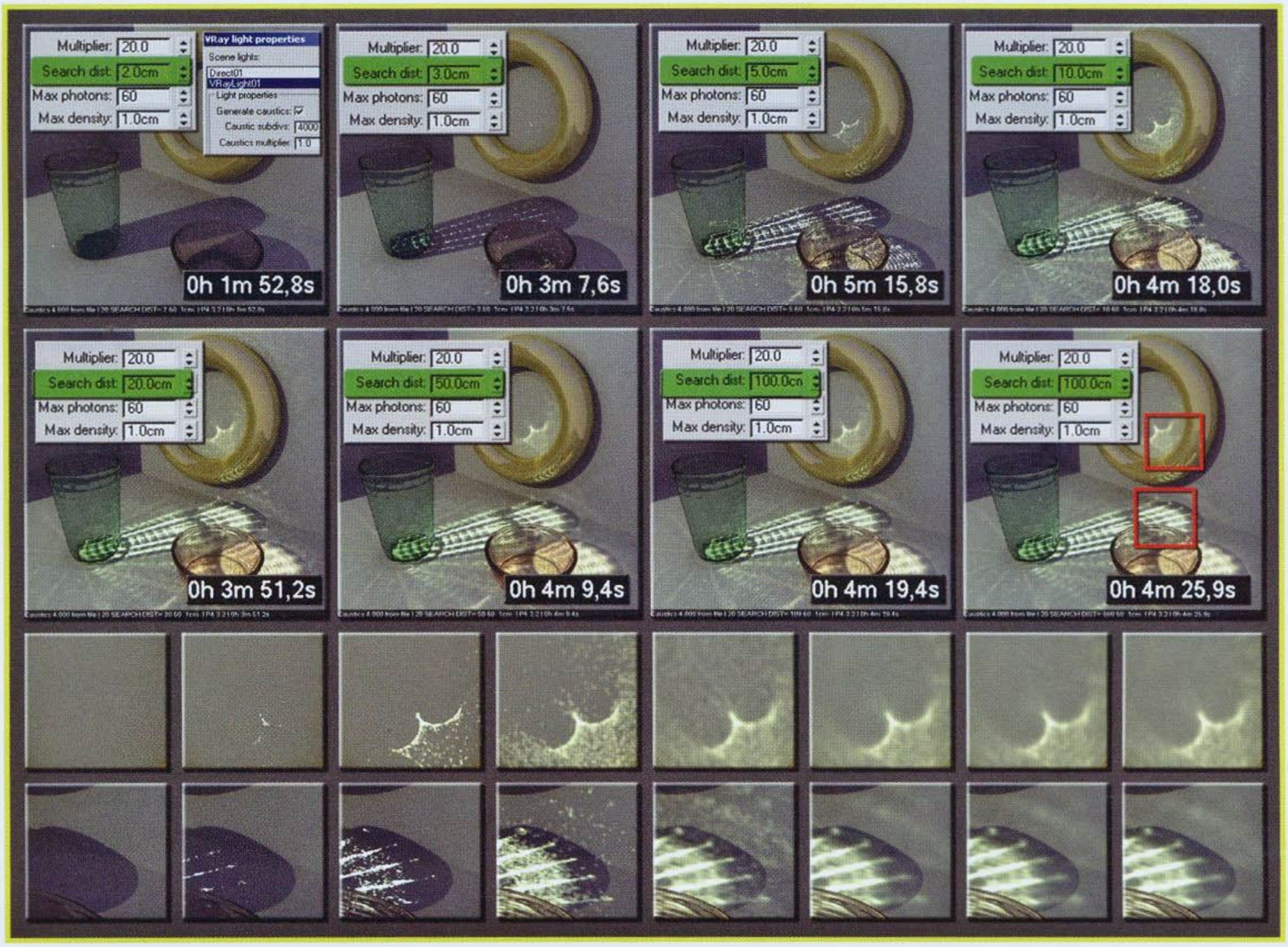


■ **Figure 4.28**
VRay: Caustics rollout in the Renderer panel.

On - activates and deactivates caustics.

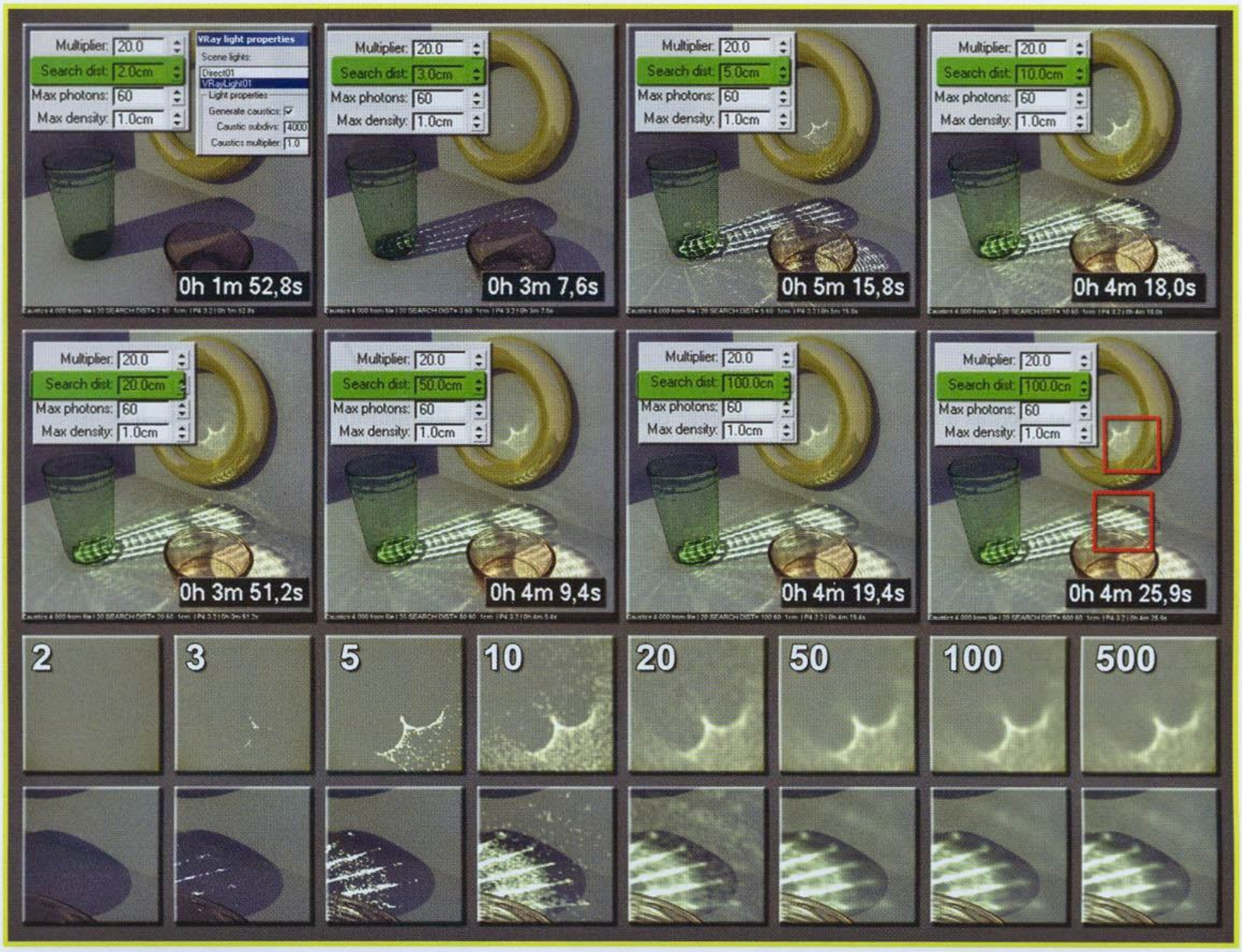
Multiplier - this multiplier controls the intensity of the caustics. It is a global value, applied to any light source in the scene and fit to be used as a caustic generator. In case one wants to modify the multiplier for each light, it is necessary to set this parameter singularly, working inside the panel ***VRay light properties Multiplier*** is cumulative, in the sense that it is added to the multiplier of each light.

Search dist. - when *VRay* emits photons, these hit a surface at some point, on which caustics form. The value ***Search dist.*** searches the photons which outline the rendering point considered, to compute caustics. In other words, the value ***Search dist.*** is the radius of a circle having as its centre the considered photon.



■ **Figure 4.29**
Variations of the parameter ***Search distance*** for modifying caustics.
--- DVD ---

The bigger this radius is, the higher the number of photons considered. This makes the caustics become more uniform, but lacking details too. Another effect is the increase in rendering time, as a higher number of photons is used. This parameter can be modified after the computation of the ***Caustic Photon Map***. In fact, one only has to load it in memory and through this it is possible to modify ***Search dist.*** without having to compute the entire map once again, saving a lot of time. In this case a low-quality ***Caustic Map*** has been used, generated by 4,000^2 photons roughly. What might be noticed is that the uniformity of caustics increases as the value of the parameter ***Search dist*** increases. This happens for the rendering time too, at the expense of the level of detail.

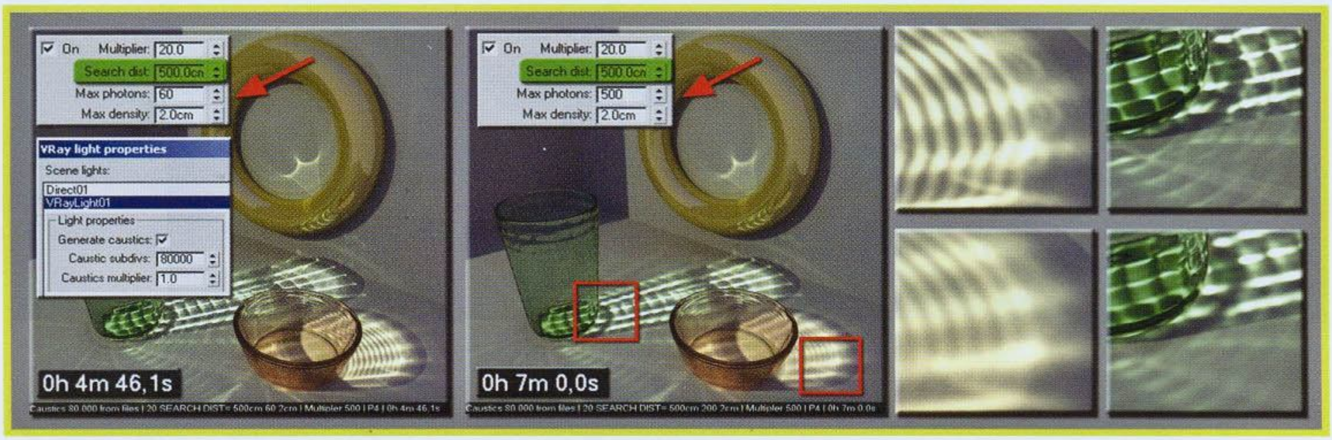


■ **Figure 4.30**
Variations of the ***Search distance*** parameter for modifying caustics, using a low-quality ***Caustic Map***.

In this second example a high-quality ***Caustic Map*** has been used, obtained after more than an hour of computation. It is to be noticed how very accurate and precise caustics have been obtained with low values of ***Search dist***. The passage from 4 to 10 leads to a considerable lack of detail, but it also gives a greater uniformity. In this case, the "boundary threshold" created by the parameter ***Max photons*** is such that, for values higher than 50, the increase of the parameter ***Search distance*** is negligible in modifying caustics.

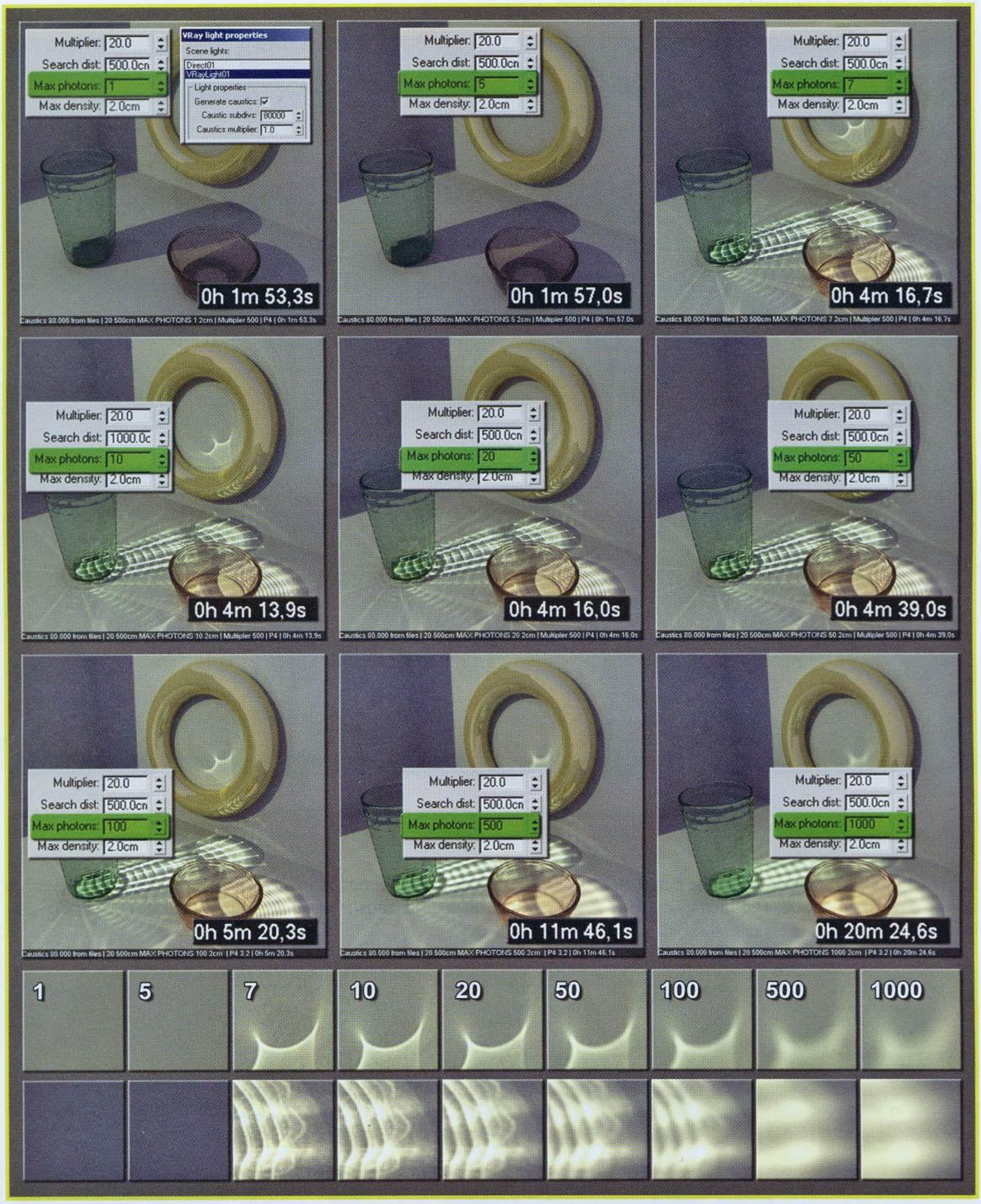
■ **Figure 4.31**
Variations of the parameter ***Search distance*** for modifying caustics using a high- quality ***Caustics Map***.

In the test above the meaning of the parameter ***Max Photons*** has been mentioned in advance. Keeping the value ***Search distance*** constant at 500, and increasing ***Max Photons*** from 60 to 500, caustics become more faded. It is clear that this last solution is not the best one: too blurry.



■ **Figure 4.32**
Variations of the parameter ***Max Photons*** keeping ***Search distance*** fixed.

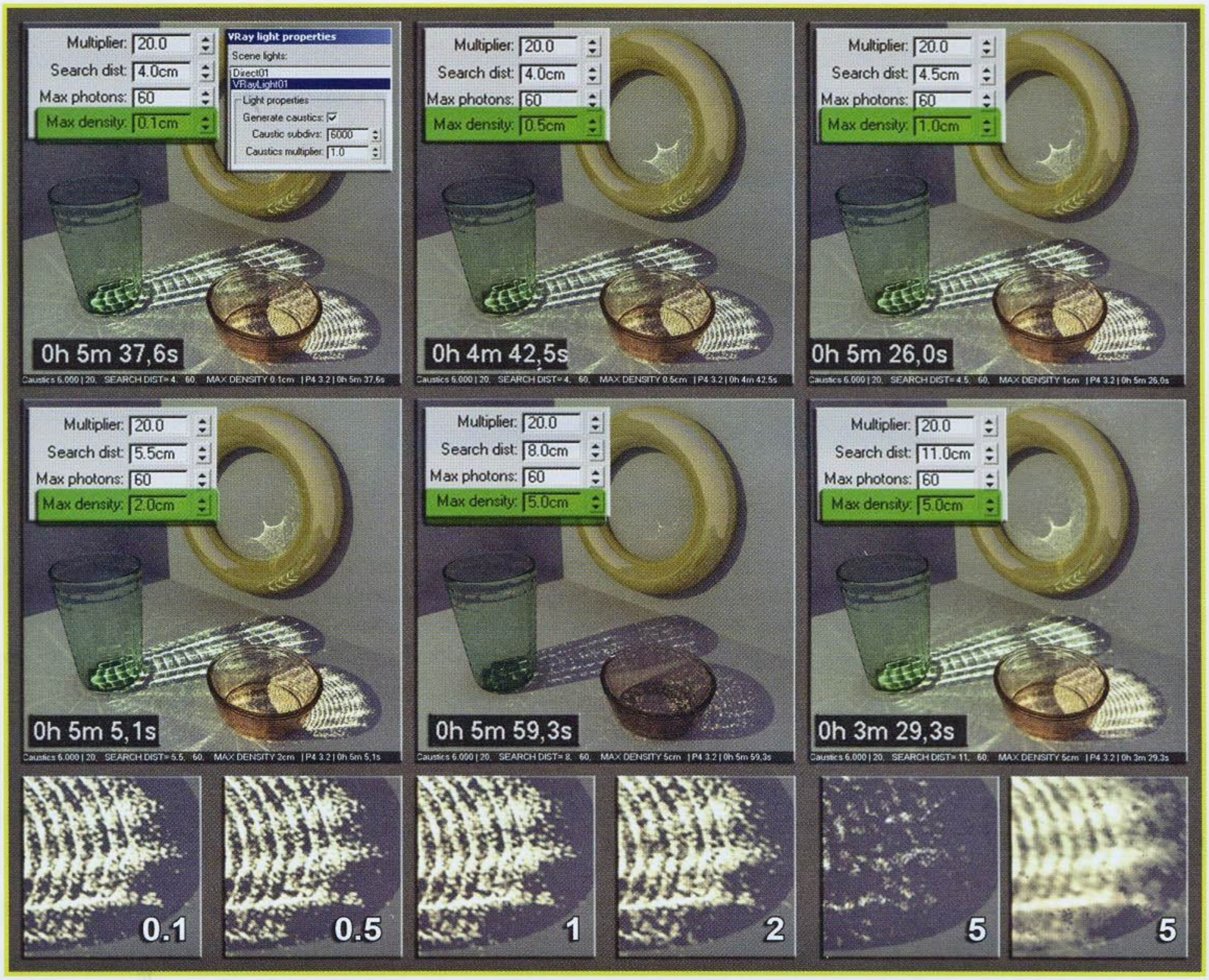
Max photons - when *VRay* attempts to compute caustics in a certain area determined by the value ***Search dist***, it physically counts the number of photons inside this perimeter. If the number of photons is higher than ***Max photons***, *VRay* only uses the number equal to the ***Max photons*** setting.



■ **Figure 4.33**
Variations of the parameter ***Max photons*** for modifying caustics.

There are essentially two effects deriving from the variation of this parameter: the first one concerns the quality of the render. The higher ***Max photons*** is, the more uniform caustics are, but with loss of details. In this case the best solution is the value 10. The second effect concerns rendering times. The higher the parameter ***Max photons*** is, the higher the amount of photons considered in the computation is. This increases the time necessary for the computation.

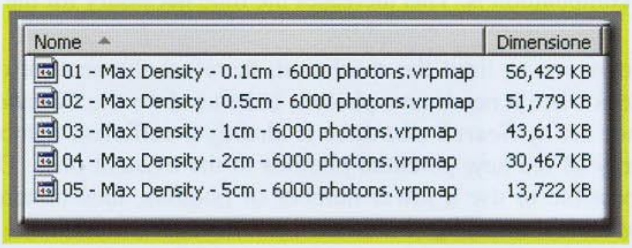
Max density - this parameter allows one to limit the resolution, therefore the amount of memory as well, of the ***Caustic Map***. The first thing *VRay* does when it needs new photons to be used for the ***Caustic Map***, is to check whether within the area defined by the distance set by ***Search dist*** there is already a sufficient number of photons. If there are already enough, *VRay* sums the energy of the new potential photons to the existent ones. Otherwise new photons are distributed. Using this option it is possible to use a lower number of photons, thus obtaining smoother results, less detailed and keeping the dimension of the ***Caustic Map*** within the limits.
In practice, ***Max density*** can be compared to the function of subdivision of the mesh referred to the Radiosity method. The lower this value is, the lower photon cells are, and the rendering is more detailed. But in this case a lot of photons are needed to obtain caustics without artefacts.



■ **Figure 4.34**
Variations of the ***Max density*** parameter for the computation of caustics.

In the test above one has tried to obtain the same quality for caustics, just changing by the value ***Max density***. To achieve this, each time the value ***Search dist*** has been "adjusted". In fact, these two values are tightly connected. Usually ***Search distance*** must be 5-10 times the ***Max density*** parameter. In such a way there is a certain area where photons are to be searched for drawing caustics. ***Max density*** works as a grid. The lower its value is, the smaller the cells are, but for a good quality a higher number of photons is necessary. The real advantage of this parameter is that it limits the dimension of the map, to avoid excessive file size. Vice versa, if the value is too high, evident artefacts are visible and caustics are approximate.

By analyzing the tests, one can deduce that the thresholds are roughly 2, for the minimum one, and 5, for the maximum one. At higher values caustics are too imprecise and lacking detail. Instead, below the value of 2, ***Max density*** does not have any effect, except it increases the size of the ***Photon Map***.

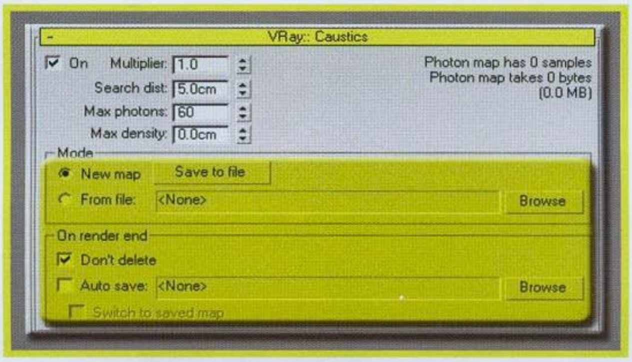


■ **Figure 4.35**
Increasing the parameter ***Max density*** the size of the map decreases considerably.

By observing the data, one can notice how the map is halved passing from a value 0.1 cm to 2 cm. To be noticed that the number of emitted photons is just 6,000 ***Caustic subdivs***.

Mode

The second part of the rollout ***VRay: Caustics*** concerns the functions managing the ***Caustics Map***.



■ **Figure 4.36**
The section of the rollout ***VRay: Caustics*** which is dedicated to saving the ***Caustics Map*** is highlighted.

Mode - all of these parameters enable the control of the ***Caustics Photon Map***.

- ***New map*** - a new map is always generated if this parameter is active. If a map has already been computed with ***Caustics Map*** and therefore is already present in memory, it is to be replaced if a new render is computed.
- ***Save to file*** - by clicking this button, the map contained in the memory is saved in a file.
- ***From files*** - when activated, it is possible to load the map previously computed from a file by clicking on ***Browse***.
- ***Browse*** - it shows the path for loading the file.

On render end

Don't delete - if active, ***VRay*** keeps the ***Caustics Map*** in the memory after the scene has been rendered, leaving one the possibility of saving it. By deactivating it, after the rendering the map is to be unavoidably lost.

Auto save - if activated, ***VRay*** automatically saves the map in a file chosen by the user, at the end of the computation of the ***Caustics Map***.

Switch to saved map - this option is available only if ***Auto save*** is active. By clicking it, at the end of computing of ***Caustics map***, ***VRay*** saves the map in a file, loading it with the option ***From files***.

Finally, a last section is present, which is not editable by the user, but gives feedback on some data concerning the ***Caustics Map***, amongst which the number of samples used and the size in MByte of the map generated or loaded in memory.

- THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK -

VRay: rQMC Sampler

INTRODUCTION

The ***Monte Carlo*** method belongs to the family of statistical methods. It is useful for solving computational problems connected to exact tests, for example methods based on combinatorial analysis. From a point of view which is not merely statistical, ***Monte Carlo*** simulations are numerical techniques used to compute integrals. Its origin dates back to the '40s, and it was born within the Metropolis project. ***John von Neumann*** and ***Stanislaw Marcin Ulam*** have been the performers of this method, while its name is referred to the famous casino. The ***Monte Carlo*** algorithm is a numerical method employed to find solutions to mathematical problems with many variables, which cannot be easily solved, for example, with an integral method. The efficiency of this method, compared to other ones, grows as the extension of the problems does. The first example of use of ***Monte Carlo*** is the experiment of the needle of Buffon, but maybe the most important use is ***Enrico Fermi***, who used, in 1930, a random method for computing the properties of neutrons.

The ***QMC*** (***Quasi Monte Carlo***) is employed in *VRay* for any function where sampling operations are involved, such as ***antialiasing***, ***DOF***, ***Global Illumination***, ***Irradiance Map***, ***Area Shadows***, ***Glossy reflection***, ***refraction***, ***SSS***, ***Motion Blur***, etc... The ***QMC*** is used to determine which samples must be used for sampling and, finally, which rays are to be used for the ongoing operation. A parameter, usually called ***Subdivs***, is always assigned to each operation. It allows one to set the quality of the sampling. In addition to the value ***Subdivs***, other parameters allow ***QMC*** management. But in this case it has been decided to put them together in a unique rollout: the ***VRay: rQMC sampler***, where the "r" stands for random.

Before going forward it is necessary to explain the definitions of ***Subdivision*** and ***Samples***. In *VRay*, ***Subdivisions*** are the unit of measurement of the maximum number of samples (rays) which *VRay* uses to compute a certain effect. The maximum number of samples is proportional to the square of the value of ***Subdivs***. For example, if ***Subdivisions*** of an unfocused reflection is 5, then *VRay* does not use more than 5x5=25 samples to compute this effect. At the most, it could use a lower quantity. The number of samples *VRay* uses to compute the different effects can be affected and determined by several factors, such as:

- ***Subdivs***, specific for each parameter. Each one has a specific spinner enabling the setting of its ***Subdivision***. Further on, all ***Subdivisions*** can be multiplied at once thanks to the option ***Global subdivs multiplier***, as will be shown soon.

- The importance of the blurry-effect value, also called ***Importance sampling***. For example, as will be shown in the chapter dedicated to materials, unfocused dark reflections require fewer samples for a render without noise than bright reflections do. As a consequence, bright pixels have a greater importance than dark ones. In other words, the further ***VRayLights*** are from the illuminated object, the higher the number of samples to be used for a correct shaded shadow must be.

- The variation, meant as noise, of samples. For example, if the samples of a ***glossy*** reflection are similar to each other on a certain surface, very few ***Subdivisions*** are needed for a clean render. Vice versa, the bigger the difference between different samples is, caused by complex reflections or ***Area Shadows*** produced by several colored lights, etc..., the greater the number of the samples must be to give a clean render. To carry this out, *VRay* observes the samples one by one while they are computed, and decides, for each new sample, if its presence is necessary or not to achieve the quality set by the user. It might also decide to use them all, at the expense of time. This sample-selection technique is named ***Early termination***. If the adjustment is very low, *VRay* does not distinguish if it is necessary or not. That is, it uses all of them.

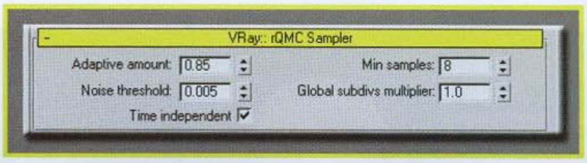
Further on it will be explained what happens when *VRay* has to compute blurry effects and how to choose the number of samples to be used for the computation of unfocused effects. *VRay* starts by carrying out two steps:

1. The first one, during which the maximum number of samples to be used for a specific effect is set. This quantity is managed by the parameter ***Subdivs*** present in each blurry option (***Subdivs*** for ***antialiasing***, ***Subdivs*** for the ***DOF***, etc...). We shall call this value N.

2. The second step, during which the minimum number of samples to be used is set. This quantity will never be less than the value set in the parameter ***Min samples*** (it will be soon explained). It also depends on the amount of adaptability of the samples managed by the parameter ***Adaptive amount***. The less the adaptability is, the higher the number of used samples is. We shall call this value M.

After having done this, *VRay* goes on computing the samples M first. For the leftover samples, that is the ones remaining from the subtraction N-M, *VRay* has to decide if they are "good" enough to be employed in the rendering or not. The parameter ***Noise threshold*** is involved for this task. It decides what is usable or not. If *VRay* decides the result is satisfying, or also if all the N samples have been processed, the final result, that is the render, is shown.

PARAMETERS



■ **Figure 4.37**
VRay: rQMC Sampler rollout located in the panel Renderer of 3ds Max.

Adaptive Amount - it controls the threshold where the techniques ***Early termination*** and ***Importance sampling*** are applied. In other words it controls the number of samples to be used for a specific blurry effect, in this case the ***glossy*** effect in the reflection of the floor.

■ **Figure 4.38**
Changes in the ***Adaptive amount***, parameter located in the rollout ***VRay: rQMC***.

With high adaptability, only samples which require a greater care are to be processed, decreasing rendering time at the expense of quality. Value 1.0 means a complete adaptation. The value 0.0 means no adaptation.

In the test which has just been carried out, one can notice the quality both of the ***glossy*** effect and the ***Area Shadows*** are improved as the parameter ***Adaptive amount*** decreases. Obviously at the expense of time. ***Antialiasing*** has been purposefully deactivated, in addition to the employment of values of ***Subdivision*** fixed to 4 (low values) both for ***Area Shadows*** and for ***glossy***, in order to better observe these two particular effects. The best result in terms of quality/time ratio is achieved with ***Adaptive amount*** around 0.5/0.6. The parameter ***Noise threshold*** has been increased appropriately. This way its influence on render quality is almost none, and sampling is committed to the ***Adaptive amount***.

Min samples - it fixes the minimum number of samples to use before the algorithm ***Early termination*** is used. High values slow down calculations and rendering as a consequence, improving the quality nevertheless. It is reminded that the algorithm ***Early termination*** allows *VRay* to use less rays compared to the number set in the parameters ***Subdivision*** (GI, ***glossy***, ***Motion blur***, etc…) in case the quality of the render is already satisfying and the rays in excess would not change it that much. The higher ***Min Samples*** is, the more samples are excluded from adaptability, obtaining a higher quality, at the expense, however, of time.

■ **Figure 4.39**
Variations of the parameter ***Min samples***.

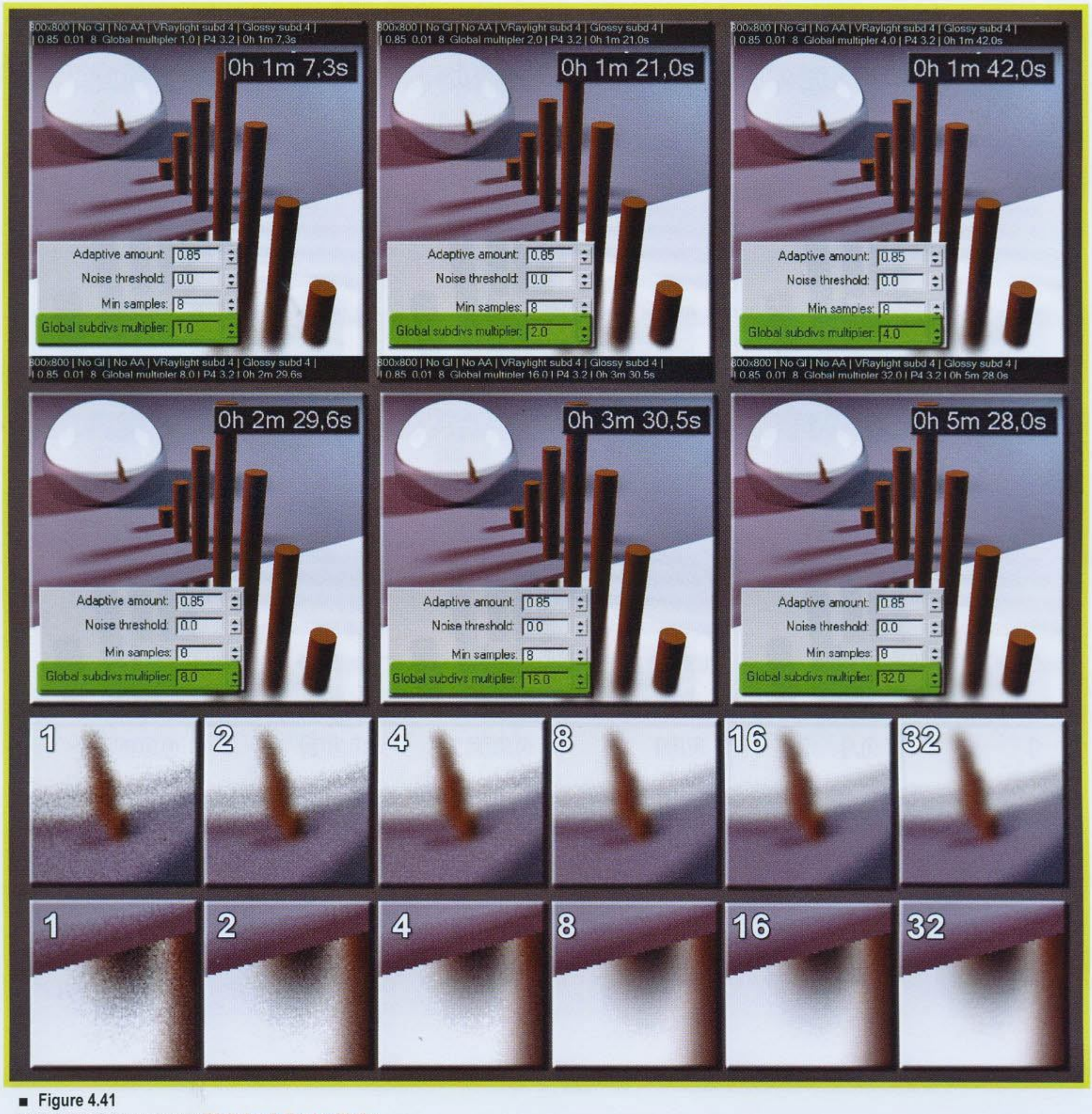
In this test, noise has been increased by raising the parameter ***Noise threshold***; after the parameter ***Min samples*** has been increased. Raising its value the quality is higher too, decreasing the general noise.

Noise threshold - parameter allowing *VRay* to decide when a sample is "good enough" for being used for rendering calculations. Practically this means a general increase or a decrease of the noise. Low values produce less noise, and high values make the noise more evident, but rendering times are lower.

■ **Figure 4.40**
Variations of the ***Noise threshold***, parameter in the rollout ***VRay: rQMC Sampler***.

In this test it is clear how general noise decreases if ***Noise threshold*** decreases. In any case, it has not been possible to remove it completely even with very low values. This is attributable to the low ***Subdivs*** both of ***Area Shadows*** and the ***glossy*** , and to the deactivation of ***antialiasing***. Usually, for high-level renders, one should never set values lower than 0.002, because rendering time would then increase to excess.

Global subdivs multiplier - this parameter multiplies almost all ***Subdivs*** parameters within ***VRay***. The only parameters not involved are: ***Light Cache***, ***Photon Map*** and the ***antialiasing Subdivs***. Instead, ***DOF***, ***Motion blur***, ***Irradiance Map***, ***QMC GI***, ***VRayLight***, ***glossy*** and reflections/refractions are affected. This way, the user is able to rapidly increase or decrease the sampling quality for all blurry parameters. It is a very quick system for setting the quality of the render or for carrying out a test-render. In fact, it is possible to use minimum values of 1, speeding up the rendering at the expense of time.



■ **Figure 4.41**
Variations of the parameter ***Global subdivs multiplier***.

Now we shall try to understand how this parameter works. ***Glossy*** and ***VRayShadow*** have ***Subdivs*** = 4. For example, in the case of ***Global subdivs multiplier*** = 16, one would obtain 16 samples x 16 multipliers = 256 samples. Thus, setting the ***Subdivs*** of ***VRayLight*** and ***glossy*** to 16, to obtain the same result just seen, the ***Global subdivs multiplier*** has to be set to 1.

EXAMPLE 1

a.	Subdivs effect	=	4	->	4	x	4	=	16	samples
b.	***Global subdivs***	....						=	16	
TOT	samples	=	a*b	=	16	x	16	=	256	samples

Which corresponds to:

a.	Subdivs effect	=	16	->	16	x	16	=	256	samples
b.	***Global subdivs***	....						=	1	
TOT	samples	=	a*b	=	256	x	1	=	256	samples

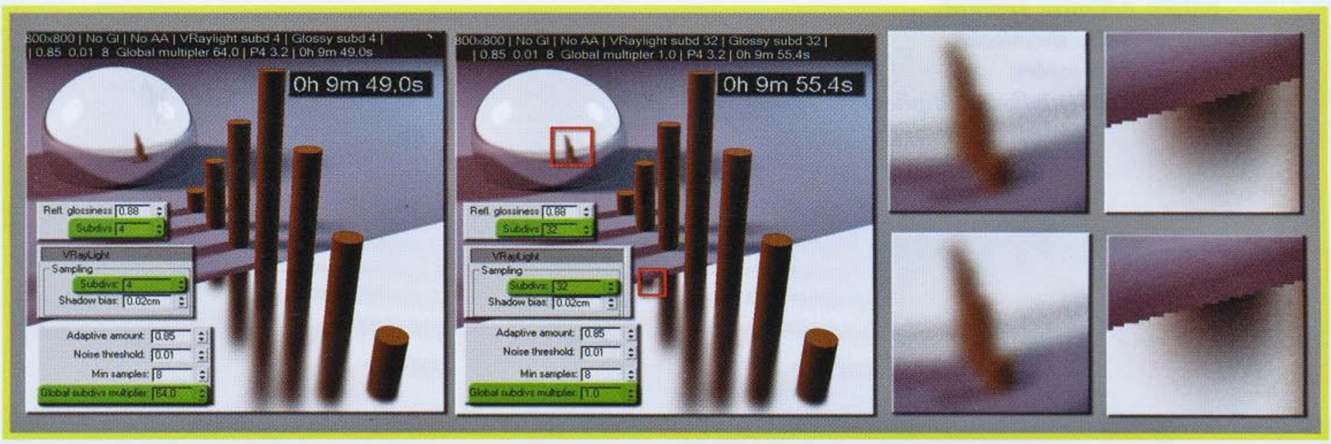
EXAMPLE 2

a.	Subdivs effect	=	4	->	4	x	4	=	16	samples
b.	***Global subdivs***	....						=	64	
TOT	samples	=	a*b	=	16	x	64	=	1024	samples

Which corresponds to:

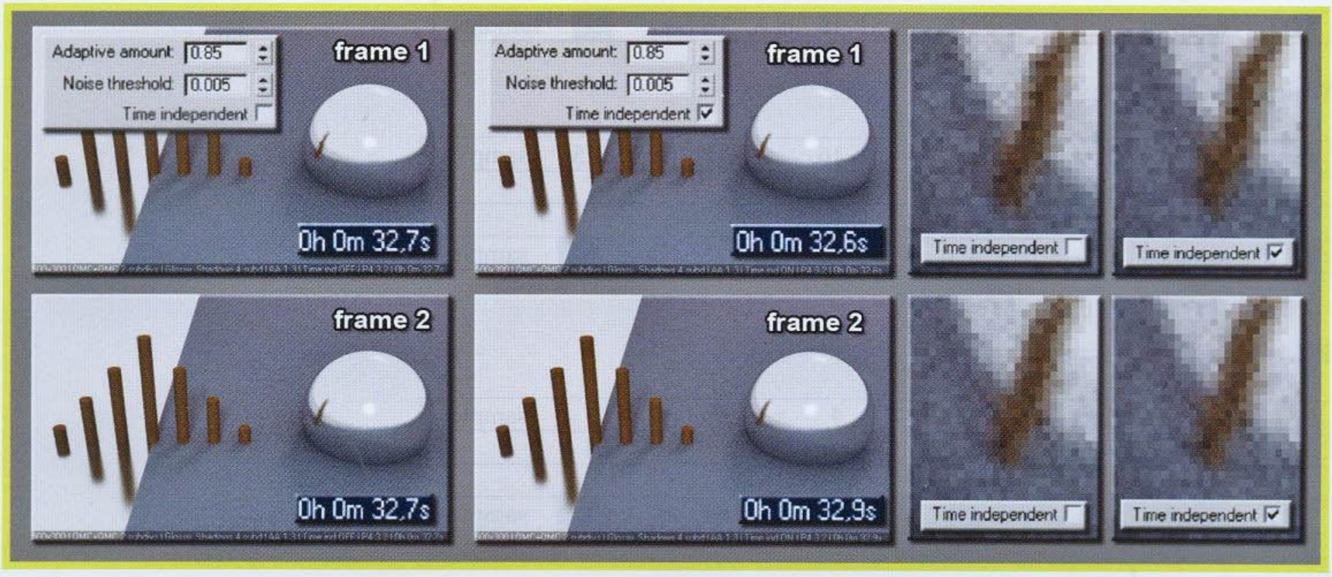
a.	Subdivs effect	=	32	->	32	x	32	=	1024	samples
b.	***Global subdivs***	....						=	1	
TOT	samples	=	a*b	=	1024	x	1	=	1024	samples

■ **Figure 4.42**
Example 1.



■ **Figure 4.43**
Example 2.

Time independent - when this function is active, the arrangement of noise in the frames of an animation remains the same, avoiding unattractive vibrations of the image. Instead, by deactivating it, it is possible to obtain the pattern of different noises between frames. Rendering the same frame a second time, both with the option ON or OFF, one obtains the same result.



■ **Figure 4.44**
Test of activation and deactivation of the parameter ***Time independent***. Unfortunately the only way to understand the use of this parameter is observing an animation. Rarely are two frames alone able to clarify its use.
- - - DVD ANIMATION - - -

What is observed by analyzing animations with ***Time independent*** disabled is a slight fluttering of the animation itself, due to the change on the pattern of the noise. It is clear that the greater the number of ***Subdivision*** of various effects is, the more the noise tends to decrease, even with the ***Time independent*** disabled.

NOTES

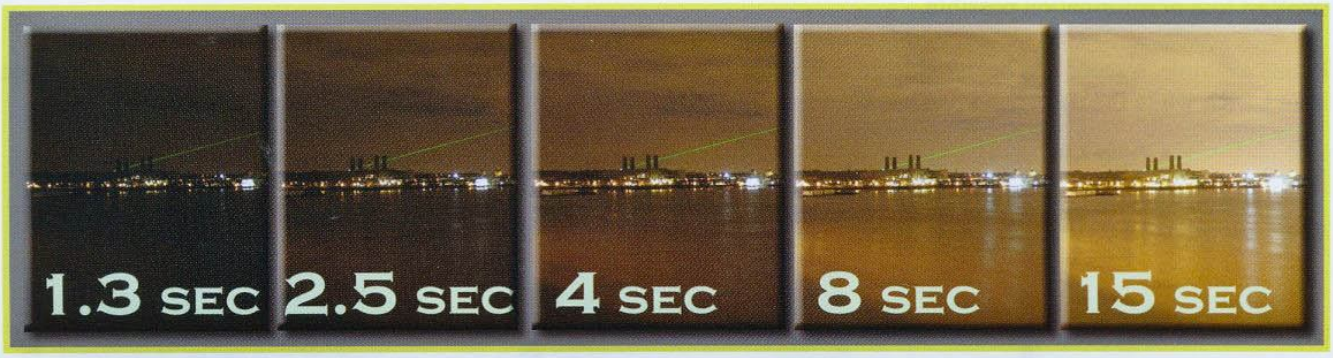
- ***Adaptive amount*** and ***Time-independent*** are the only parameters considered by the ***PPT***. It is not advisable to use ***Adaptive amount*** with value 0.0, as it would block the rendering.
- To obtain renderings with low noise, but with a greater control on each value of the blurry, it is advisable to use low values of ***Adaptive amount*** in the place of low values of ***Noise threshold***. In this case one obtains a lower adaptability due to the low value of ***Amount***, but with a greater control on the ***Subdivs*** of single parameters.
- Summarizing, for higher quality it is necessary to:

  - reduce ***Adaptive amount***;
  - reduce ***Noise threshold***;
  - raise ***Min samples***;
  - raise ***Global subdivs multiplier***.

VRay: Color Mapping

INTRODUCTION

In photography, the term exposure means the time during which the shutter of the camera remains open in the snapshot of a photo. More often in technical slang, the same word means the total amount of light which reaches the film (or the sensor in case of digital cameras) in that time. The exposure is measured in EV (exposure values) and is set with the exposure meter.



■ **Figure 4.45**
Mathematically, exposure is defined by this law: exposure= luminous intensity x time.

The reason for the presence of different methods of color mapping inside of *VRay* is the possibility of modifying renders as happens for a usual camera. If one walks into a room, lit only by a small opening, the human eye compensates the little amount of light by widening itself, allowing a correct view of the room. Using a camera, problems such as under and overexposure can occur. It all depends on the settings the user has fixed during the snapshot. With *VRay* the same thing happens. Rendering indoor scenes lit by outdoor lights without adding some extra lights, which are not physically correct, can be difficult if not using color mapping methods.

Now the next step is clarifying the idea of floating point images, such as .hdr, .exr, .rla. These images are different from the common ones, such as .bmp, .tga, .jpg etc... These last ones are defined in the RGB color space. The RGB assigns to each pixel an intensity value, included between 0 (black) and 255 (white) for each RGB component. These channels are red, green and blue. Each channel can be represented by 256 shadings, for a total number of 16,777,216 colors (256 x 256 x 256). So images are defined by 24 bits ($2^8 \times 2^8 \times 2^8 = 2^{24}$).
In the case of a monochromatic image, where color is either black or white, only two colors are available. To define this characteristic the concept of Color Depth is introduced. With this term one means the amount of bits needed to represent the color of a single pixel in an image bitmap. This idea is also known as **bit(s) per pixel** or **bpp**. Monochromatic images have a color depth equal to 1 bpp. That is, the pixel can be white or black (on/off). Here below the most common color depth formats:

1 bpp

MONO	=	2^1	=	**2**	colors

2 bpp

CGA	=	2^2	=	**4**	colors

4 bpp

EGA	=	2^4	=	**16**	colors

8 bpp

VGA	=	2^8	=	**256**	colors

The depth model Truecolor allows one to reproduce images with good accuracy, reaching up to 16.7 millions of different colors. With this depth, 8 bits are used for representing the red, 8 for green, and 8 for blue: that is, in total, **24 bits** for each pixel. The $2^8 = 256$ intensity levels for each channel combine together producing 16,777,216 colors (256 x 256 x 256). The term "32-bits color" might be misleading, because in this case there are not 2^{32} distinct colors, but there is truecolor with 8 bits added which might not be used or useful to the alpha-channel codification.

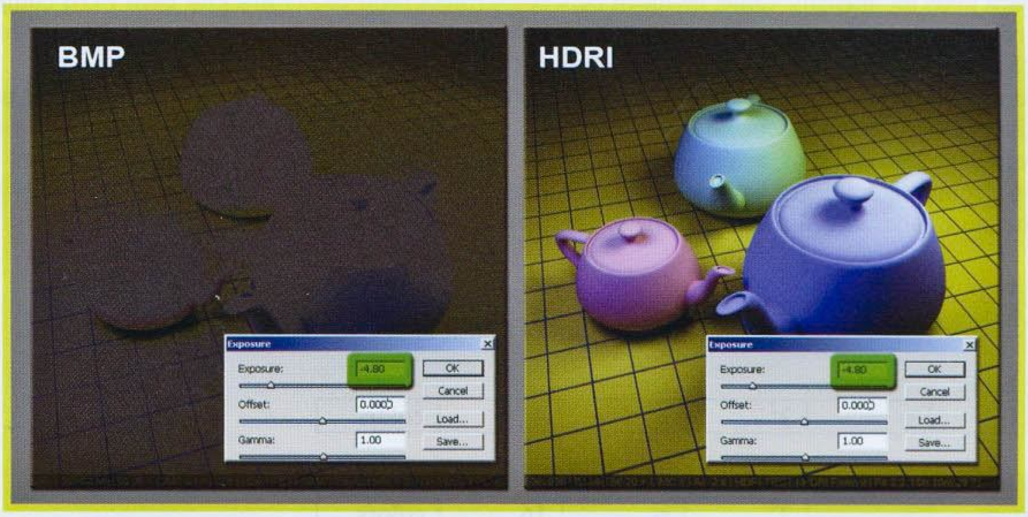
In comparison to LDRI images (**L**ow **D**ynamic **R**ange **I**mages), that is the usual .bmp, .jpg, etc… where the intensity values of pixels are treated as natural numbers quantized to 8 bits, the HDRI (High **D**ynamic **R**ange **I**mages) are images capturing the whole luminosity range of the real world, starting from the dark 0.0 up to sun light, 100,000. To make this possible for each channel the information is coded by real values, usually 32-bit values with floating-point.

The common employment of HDRI images has lead to the problem of representing them in an output device having a narrow dynamic range compared to the one of the image itself. The mapping process is called color mapping. In other words the color mapping maps the high dynamic range of the image into the narrower one of the screen with the smallest loss of information possible.

In this example, a scene has been rendered deliberately increasing the multiplier of the two ***VRayLights***. The left image is an LDRI, while the right-hand one has been saved in HDRI format. They appear quite similar.

■ **Figure 4.46**
Images of the same render saved in the formats .bmp and .hdr. Apparently they do not seem to be different…

Later their exposure was changed in Photoshop thanks to the Exposure option.



■ **Figure 4.47**
…but now they do not seem the same at all!

By changing the exposure in the HDRI image by means of Photoshop, as happens for all the 24-bits images, the white pixels RGB = [255, 255, 255] turn to grey. This happens because the pixel does not have other information in addition to the RGB space one.

Contrarily, changing the HDRI image, the result is completely different. This happens because in this case pixels contain much more information than the RGB ones. This information is "hidden" inside the pixel, and is not visible on a common screen.

By saving the image in a LDRI format, pixels with values higher than RGB = [255, 255, 255] are automatically clamped bringing them to the maximum values possible, that is white. This does not happen in HDRI images. Even though pixels seem white on the screen, a lot more information is contained about the whole range of luminosity of the real world, and therefore values higher than RGB = [255, 255, 255].

PARAMETERS

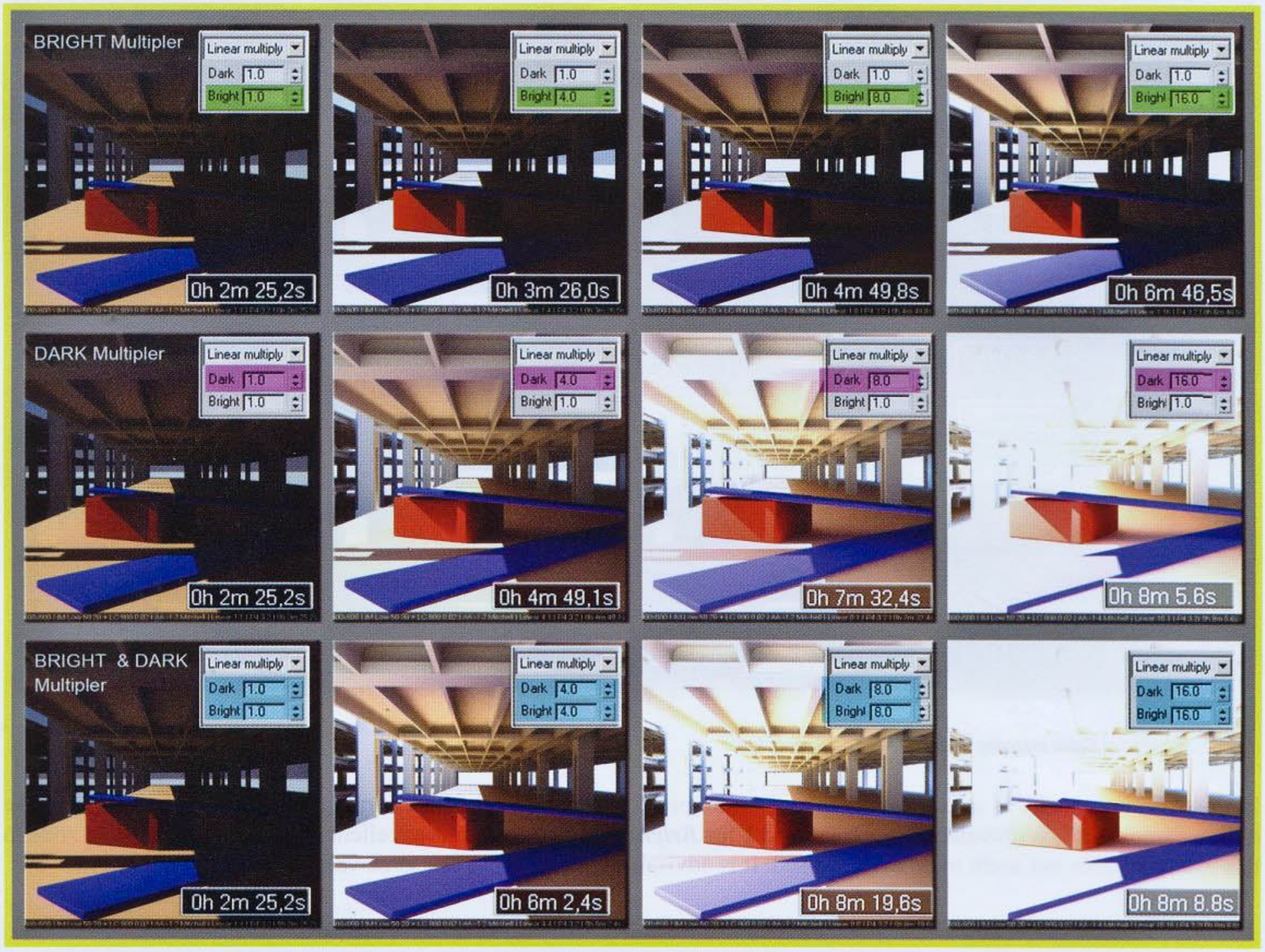
Color Mapping is a process allowing the change of colors of the final render. Five kinds of mapping exist and for each one, there are two parameters:

Dark multipiler - it allows one to brighten or darken dark pixels.

Bright multipiler - it brightens or darkens bright pixels.

These two values can be used in all five kinds of ***Color mapping***. Here below several tests are carried out, each one representing a mode, and changing each time first the ***Dark***, then the ***Bright***, and at last both of them.

Linear Multiply - it multiplies each pixel of the image in order with its luminosity. Color components of the pixels being too bright, that is greater than 255 or lower than 0, are to be cut off. As a consequence there might be burnings near very bright light sources.



■ **Figure 4.48**
Some examples of ***Color mapping*** using ***Linear*** mode.

The extreme importance of ***Color mapping*** can be noticed. Not in one of these images either the ***skylight*** intensity or the Direct Light, simulating the sun, has been changed. Only by modifying the values ***Dark*** and ***Bright multiplier*** strong differences in luminosity can be obtained.
The ***Bright multiplier*** allows one to change the luminosity of bright areas. In this example these areas are mostly found on the floor. Increasing the value, the floor brightens, until it becomes completely white.
The ***Dark multiplier*** is definitely the most important parameter. It increases the luminosity of the dark areas. With its value fixed to 4 the floor still maintains a slight saturation of the color, while internal darker areas are brighter. With the value fixed to 8, enough for an excellent illumination of dark areas, a complete burning of the areas directly illuminated by the light is also obtained. The scene is completely overexposed if the value is fixed to 16. By increasing both parameters, the effects of the ***Dark*** and the ***Bright multiplier*** are summed. By observing rendering times another observation springs to mind. The more the scene is lightened, the higher is the time needed for the render. The reason is that the light has much more energy and therefore needs more calculations to be achieved.

Exponential - this mode saturates the colors of each pixel in according to its luminosity. It is useful because it is able to avoid burning near the light sources, as well as allowing a greater light depth. This process does not clamp colors with luminosity higher than 255. Instead, their complete desaturation takes place. By using the option ***Clamp output***, as is to be explained soon, *VRay* clamps colors higher than the RGB = [255, 255, 255]. This happens because the RGB mode does not allow one to represent values above this threshold. The ***Exponential mode*** picks up all the ***Clamped*** values, scaling their intensity in accordance to their luminosity within the range RGB = [0, 0, 0] or RGB=[255,255,255]. The result is an improved distribution of light. On the contrary, the colors already included in this range are to be modified by the ***Exponential Color mapping***.

■ **Figure 4.49**
Some examples of ***Color mapping*** by using the ***Exponential*** mode.

In this case luminosity is improved compared to the previous example: evident burning is no longer present, and the light is "smoother". With values 8 for ***Dark*** and 1 for ***Bright*** the illumination is excellent and the floor, a critical zone in this example, does not seem to be overexposed. It is advisable to always use the same values for ***Dark*** and ***Bright***.

As far as the ***Linear*** mode is concerned, renders with value 16 are better, but the brown paint of the floor has disappeared, meaning that there is overexposure. Times do not exceed 4 minutes. This happens because particularly luminous zones are not present. The only trouble is a general decrease of the saturation level. It is not a particularly dangerous phenomenon in this case.

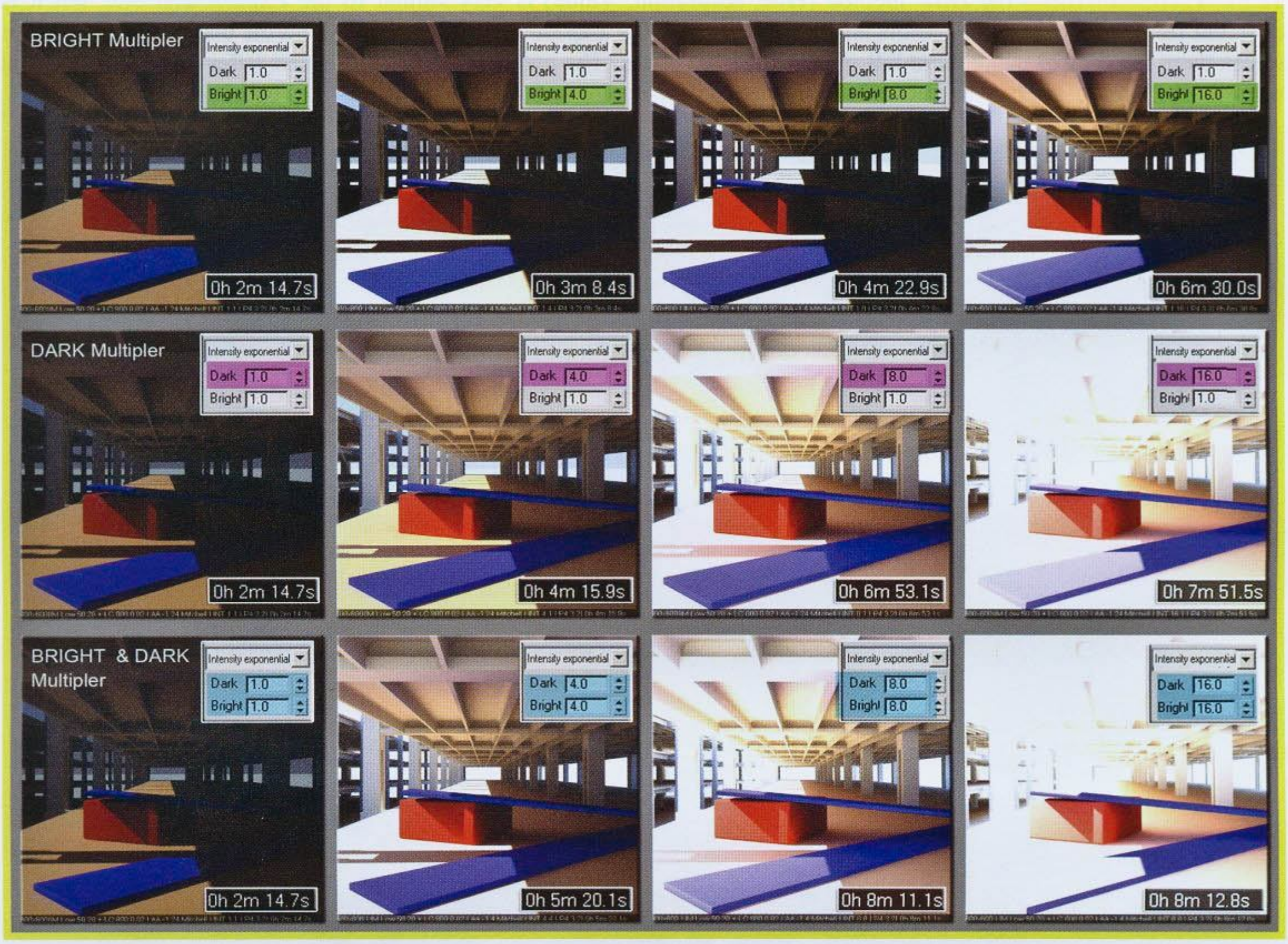
HSV Exponential - the ***HSV*** method (**H**ue, **S**aturation, **V**alue) is very similar to the ***Exponential*** one, but it is different from it in maintaining colors and saturation. Like the ***Exponential*** mode, it picks up the clamped values and it scales them in order to make them fall within the RGB space. The essential difference is that ***VRay*** takes into account not only the luminosity, but the hue and the saturation of the color of the pixel as well for this operation.

■ **Figure 4.50**
Some examples of ***Color Mapping*** using the ***HSV Exponential*** mode.

In this case one obtains a very saturated image compared to the other ones. Maybe there is even excessive saturation. Values higher than 8 cause problems. The shadow of the left pillar should be dark, and in any case not white as happens with ***Dark*** and ***Bright multiplier*** set to 16.

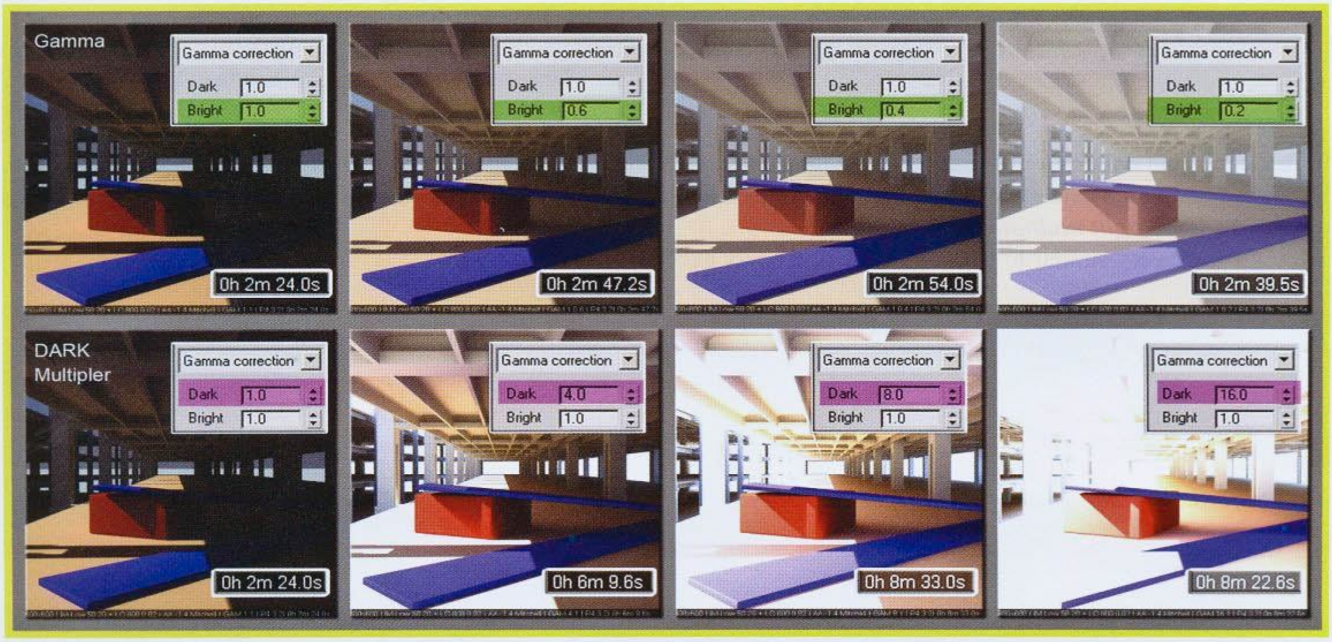
Timespans are very similar to previous ones, and never exceed 4 min. and 12 sec.

Intensity esponential - it is very similar to the ***HSV exponential*** method. Therefore, it preserves the ratio among the color components red/green/blue. The difference is it does not carry out the conversion RGB -> HSV -> RGB which may cause a loss of precision, but directly provides the result. In the example below, this kind of ***Color mapping*** does not guarantee the best illumination.



■ **Figure 4.51**
Some examples of ***Color mapping*** using the ***Intensity exponential*** mode.

Gamma correction - this method allows one to correct the gamma of the screen.



■ **Figure 4.52**
Some examples of ***Color mapping*** using the ***Gamma correction*** mode.

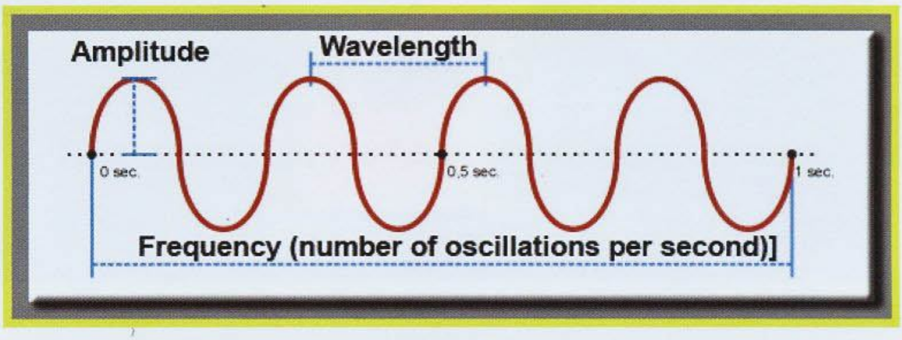
In this case the ***Dark multiplier*** simply represents the general multiplier of light, whereas the ***Bright multiplier*** shows the current screen gamma. Here is an example: if ones CRT gamma is 2.2, one has to set the parameter ***Bright multiplier*** = 0.454, that is 1/2.2, for adjusting it. Usually PC screens have gamma 2.2, whereas for Macintosh PCs the range is 1.8.
Many concepts, much more complex than a simple division, rotate around the term "gamma". These subjects are to be discussed soon. In the meantime, looking at this test, a feature can be observed. If one changes just the parameter ***Dark multiplier*** or uses the method ***Linear multiply*** with the same values of ***Dark*** and ***Bright multiplier*** the same result is obtained.

■ **Figure 4.53**
Comparison between the ***Color mapping Gamma*** mode and ***Linear*** mode. There is no difference: they are exactly alike.

At the moment, as a beta version is currently being used for *VRay*, the value ***Bright*** has not been adjusted yet. In fact, instead of it the voice ***1/gamma*** should be seen.
Of course the idea of gamma can seem incomprehensible, but soon the delicate mechanism of reproduction of the light via digital devices is to be treated. This leads to concepts such as gamma, gamut, color management, color profiles and work spaces.

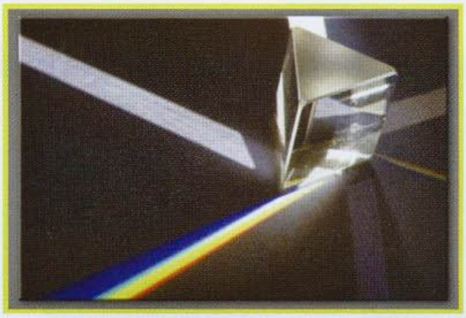
THE LIGHT

From a physical point of view, light is electromagnetic radiation. To put it simply, it is a wave propagating through space at the maximum speed possible in the universe, which is precisely the speed of light: 300,000 kilometres per second. How is this wave comprised? Exactly as all other waves are. It has crests and dips allowing one to define the three main measurements: length, amplitude, and frequency. The wavelength is the distance between two successive crests, while the amplitude is the distance between a crest and the median plane intersecting the wave; finally, the frequency is the number of oscillations the wave does per unit of time.



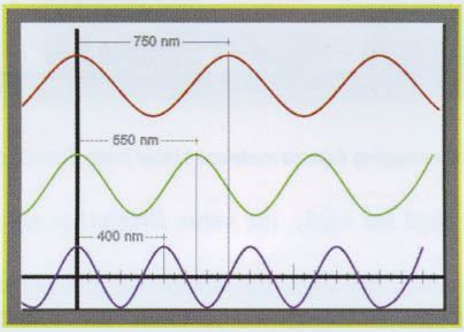
■ **Figure 4.54**
The velocity V, the wavelength λ, and the frequency f of a wave are connected by the formula $f=V/\lambda$.

The decomposition of a solar beam in colors was achieved by Newton in January 1666 in his house in Woolsthorpe, Lincolnshire. The phenomenon he described, known as light dispersion, is easy to observe. If in a dark room one makes a ray of light hit a prism, making it pass through a narrow slit and positioning a screen on the other side of the prism, a luminous band containing all the colors appears (not a white line). Although a clear distinction from one color to the next does not exist, Newton marked the most evident ones: red, orange, yellow, green, blue, indigo, and violet. Newton called this colored band **spectrum**. He deduced from this experiment that white light is a mixture of different colors, and the prism is able to separate them because of the different refraction number of the glass for each color.



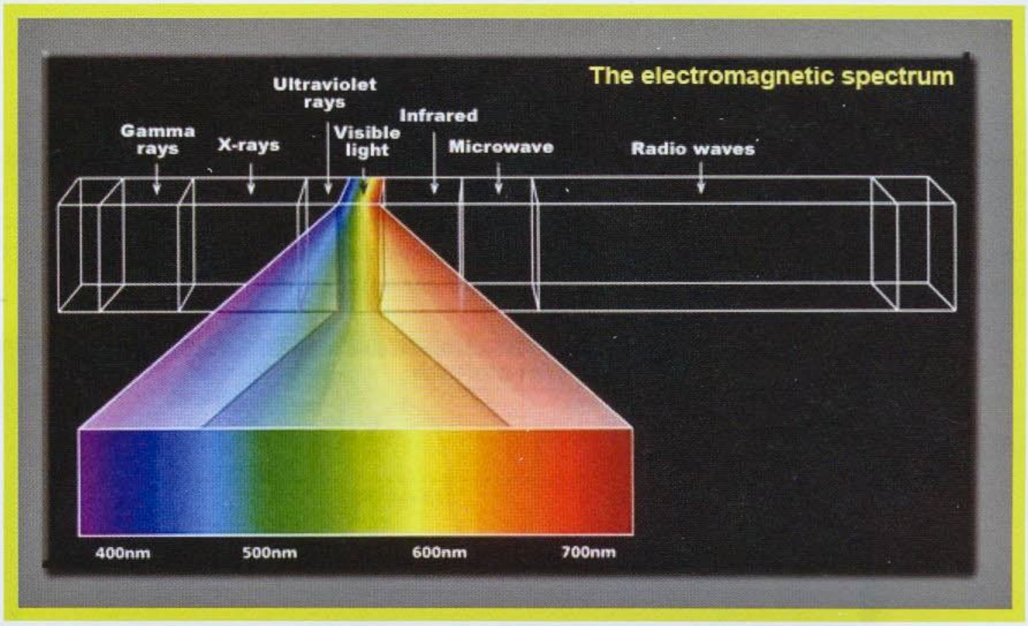
■ **Figure 4.55**
Colors are not generated by the prism. The prism is just used to separate them, carrying out the same job that small drops of water do when the rainbow appears.

The physic stimulus generating the perception of colors, light, can be precisely described. Nowadays it is known that light is a special type of electromagnetic energy moving in wave radiations. An electromagnetic radiation might be in just one wavelength, and is called, in that case, monochromatic. But usually it is a mixture of several radiations of different wavelength and energy.
Visible light has wavelengths between 380 and 780 nm (1 nanometre = 10^-9 m). The human eye perceives single monochromatic radiations as colors. A person with a normal vision of colors identifies light with a high wavelength (700 nm) as red or orange, the one with medium wavelength (500 nm) as yellow or green, and the one with very short wavelength (400 nm) as blue or violet. A certain feeling of the color corresponds to each wavelength.

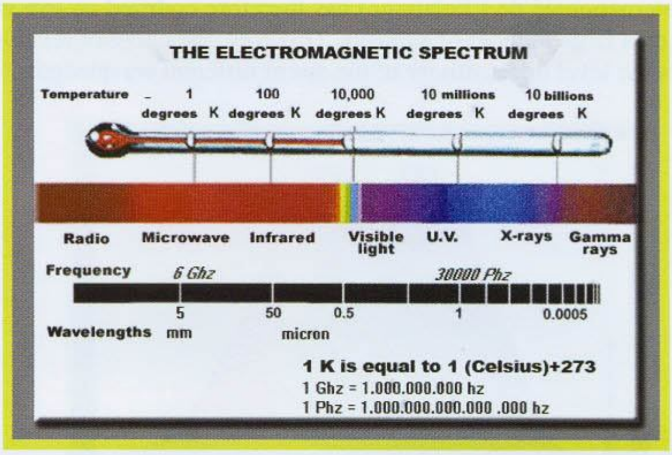


■ **Figure 4.56**
The three main wavelengths of visible light.

If one puts all the monochromatic visible radiations in order, with their wavelength and the corresponding perceived colors, the spectrum of the colors can be built. In addition, as happens for the sound, some waves beyond the possibility of human perception exist.



■ **Figure 4.57**
The whole range of electromagnetic radiations, classified in order with their frequency, composes the electromagnetic spectrum.



■ Figure 4.58
The range of variation of electromagnetic wavelengths is so wide that it is necessary to use particular prefixes before the symbol of metre (m), m (milli)=0.001 m; μ (micro)=0.000001 m; n (nano)=0.000000001 m; p (pico)=0.000000000001 m. For the same reason other prefixes are used for the frequency before its basic unit, the Hertz (Hz), meaning one cycle per second. The prefixes are: k (kilo)=1000; M (mega)=1000000; G (giga)=1000000000=10^9, and P (peta)=1000000000000000=10^{15}.

In order to have an idea of the order, let's consider the example of red light. A radiation having a wavelength of 700 nm, perceived by human's eye in normal conditions as red, is a wave in which two consecutive crests are distant 700 x 10^{-9} metres: just to have an idea of how small this distance is for humans, imagine that a million of nanometres form just a millimetre!
Even more amazing, if compared to the sizes we are normally used to, are numbers related to the frequencies of visible light. It is known that informatics technology has recently developed instruments to produce processors able to work at the speed of a Gigahertz (GHz). A processor working at 1 GHz achieves one billion cycles of complete computing per second! But at the same time, that is in one second, an electromagnetic wave with wavelength of 700 nm (what for humans is red light) achieves 428,570 billions oscillations! In other words this wave has a frequency 428,570 times higher than a 1 GHz-processor! Thus you can imagine what progress would derive for computing technology if it were possible to build processors using light rather than transistors for computing. It has probably often happened that you touched something you believed cold, getting burned. Why does this happen? Because eyes are not able to detect heat. Instead, the skin perceives it. The heating radiation deriving from an object is not noticeable for the eye. There are many things that eyes are not able to perceive. Any kind of radiation propagates through space: radio waves, infrared, ultraviolet, X-rays, gamma-rays and so on. All of this perfectly invisible. Eyes perceive only what goes from the red to the violet: it is the spectrum of visible light, the rainbow. But visible light is a minimum part of the wide electromagnetic spectrum.

Infrared radiation was discovered in the 19th century. Infrared rays cannot be seen, but can be photographed. Infrared pictures are nowadays one of the most interesting investigation mediums for human body, forests, waters and universe study. Any body producing heat produces infrared rays. These waves are similar to visible ones, they travel with the same speed, 300,000 km/s, but they have bigger wavelengths and lower frequency. For example, different temperatures on the face are made visible with different colors: the top of the nose is slightly blue, meaning that it has been photographed when it was quite cold, while under the eyes you might notice two white stains, which have higher temperatures. These images, named thermographies, are used in medicine. A house photographed in the infrared marks the losses of heat: badly isolated windows appear red.

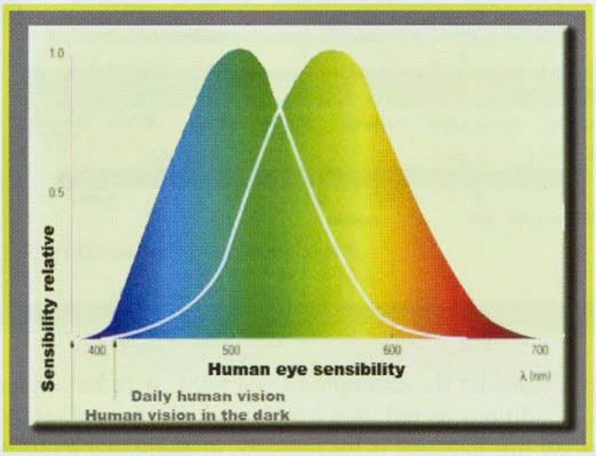
Ultraviolet radiation was discovered in 1802. As the name says, it is positioned just above the violet band. It has higher frequency than the infrared, and thus higher energy. Ultraviolet is invisible as well. It might be noticed after a prolonged exposure to sun: increasing the exposure time it can cause sunburn. Nowadays it is known that ultraviolet can be extremely dangerous, causing cancers to the skin.

X-rays have even more energy and are used both as diagnostic devices (radiographies), and healing devices against cancers. In modern astronomy, X-ray astronomy studies X-rays coming from the Sun, or far stars, or galaxies. They were discovered in 1895 by Rontgen and quickly used to obtain images of the human's body interior.

Much more energetic and dangerous are gamma-rays, coming from radioactive substances. Below infrared, thus among the longest waves, of all the radio waves to be found, with their different frequencies. One hundred years ago, Marconi used them for first for transmitting messages.

Human vision, perfect for its purpose, is able to see just a narrow window of the whole spectrum: what is needed. For the rest, human genius has built appropriate detectors. This way the human being has become able to see "everything"! But this does not mean that human beings are limited. Human eyes and those of other living beings have adapted to the types of luminous waves mostly present in their environment.

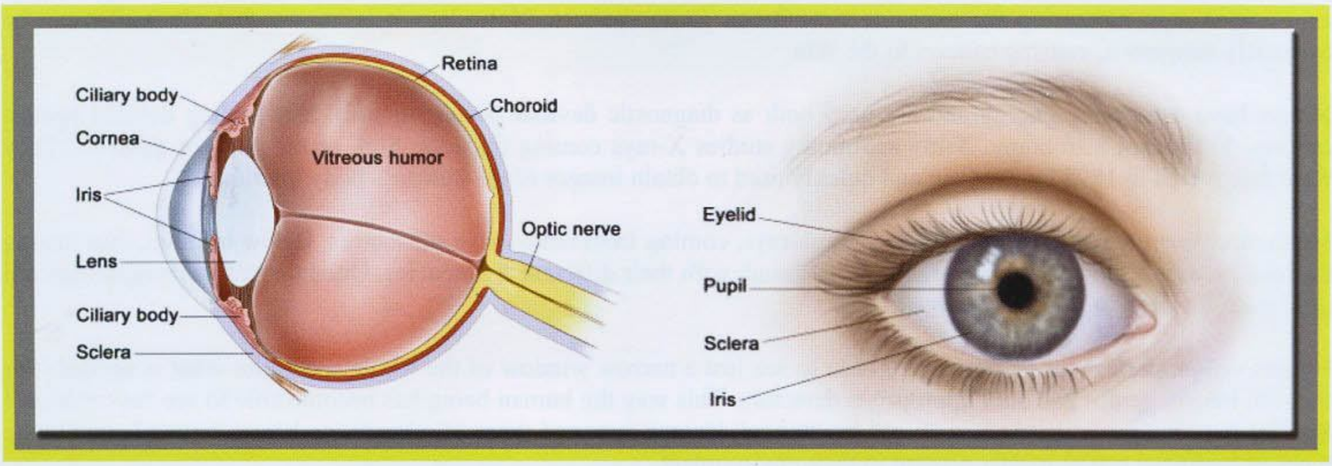
The CIE (Commission Internationale de l'Eclairage) has therefore codified an eye with average sensibility, result of a statistic work carried out on a large number of subjects. This way, the curve of relative visibility, shown here below, has been defined. It represents the level of sensitivity of the eye at different wavelengths.



■ **Figure 4.59**
Eye sensitivity is not constant for different wavelengths. It is greater for green, where the eye is much more able to distinguish tones than for other colors.

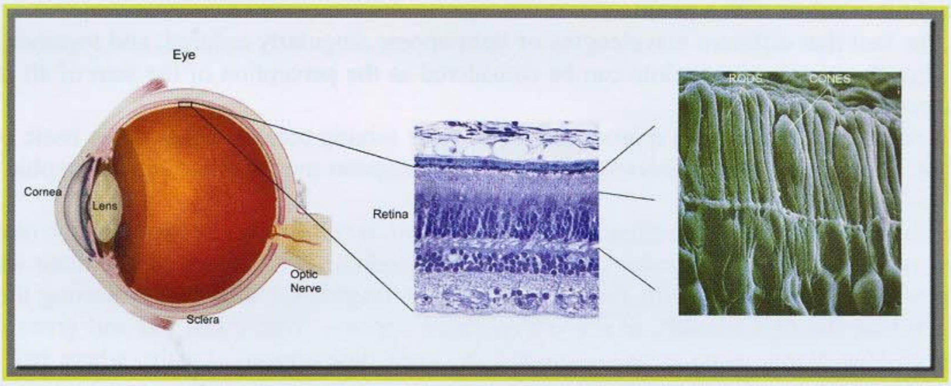
The system of human vision is correctly described if considered as the "whole complex of eyes-brain", as a vast number of processes allowing sight is carried out in specific areas of the brain.
On the other hand, in the world of digital-photography, this is not surprising: what would digicams be without the processor, which combines and computes all signals arriving from sensors? Thus, human vision is made up of:

- A system of muscles and tendons able to rotate lenses in all directions in a range of a dozen of degrees, automatically following a moving object and preserving the alignment between the two lenses to allow stereoscopic vision.
- Two equal lenses, the eyeballs, each one having an automatic hydro-mechanic system for the cleanup of the frontal element (eyelid, eyelashes, tear system) and an optical scheme of two lenses in two groups with automatic diaphragm located between the two elements in the cornea, crystalline lens, and the iris working as a diaphragm); the focal length is roughly 15mm, and the maximum opening roughly f/2.
- Two chambers full of several fluids for transmission and compensation able to keep eye characteristics stable when pressure and temperature change (vitreous humour, aqueous humour).
- A multilayer electromagnetic sensor having the same curvature of the image, and thus flatness of field (retina). Each layer has a specific function: filter (superficial blood net), sensors for low brightness (rods), sensors for high brightness (cones). This sensor has a high resolution area a few millimetres wide (fovea), provided by several dozens million sensitive elements (pixels for a digicam), a wide surrounding area, mainly used for "lateral vision", and movement perception.
- Several cables for the transmission of received signals (optical nerves).
- A heuristic processing system provided by exceptional features, with dynamic compensation tables updating in real-time, historic memory, parallel subsystems for the processing of images according to the elements composing them (sides, surfaces, shapes, etc...), spatial and temporal filtering, dynamic adaptation of color-balance, of luminosity, and contrast.



■ **Figure 4.60**
Anatomic scheme of a human eye.

Thus, visible electromagnetic radiations pass through the cornea, where they are refracted for the first time thanks to the aqueous humour, and are partially cut by the iris, encountering the crystalline lens. This element is able to focus rays, and its curvature is constantly changed by apposite muscles.
After this second refraction, rays intersect in the focal point positioned within the ocular cavity, full of vitreous humour, forming an image at the back of the eye, the retina. Light then passes through a thick net of very thin blood-vessels and encounters nerve endings of sensible cells, cones and rods. These transform electromagnetic waves in electric pulses which get to the brain, where the processing takes place, through the optical nerve.



■ **Figure 4.61**
Anatomic scheme and microscope image of cones and rods.

Not long ago, in the 19th century, it was discovered that retina sensors had different sensitivity features; in particular, rods have high sensitivity to all "color" lights, and allow vision with low brightness too, such as for example star-light; cones have have less sensitivity and are moreover divided in three subtypes: there are cones sensitive to the lower third of the spectrum, that is from deep red to yellow, called "red-cones", other ones sensible almost only in the central third of the spectrum, from orange to green/bluish, called "green-cones", and lastly other ones sensible to the outermost third of the visible range, that is from green/bluish to extreme violet. Cones are prevalent in the centre of the retina (macula lutea), while rods are denser in the outermost areas; for this reason it is suggested to look in a telescope keeping the eye oblique. In this way the image projects in an area of the retina dense of rods.

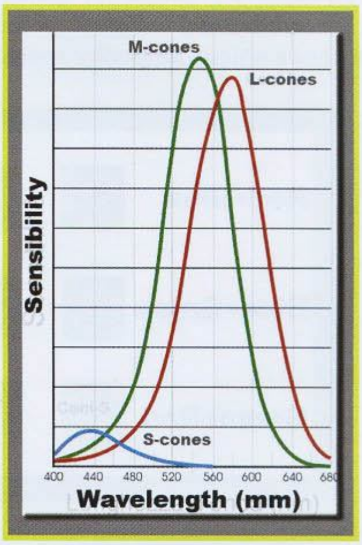
Comment: when we talk about "lower" or "superior" third of the spectrum, we refer to frequencies; many images show the spectrum as a function of the wavelength, which is the inverse of the frequency, with the blue occupying the initial part of the graph and the red the final part. In fact:

V= f/λ, where:

V= velocity.
f = frequency, number of crests passing per unit time, measured in Hz. 1 Hz = 1 cycle per second.
λ = wavelength, measured in nm.

Red frequency: 4×10^{14} Hz
Blue frequency: 7.5×10^{14} Hz

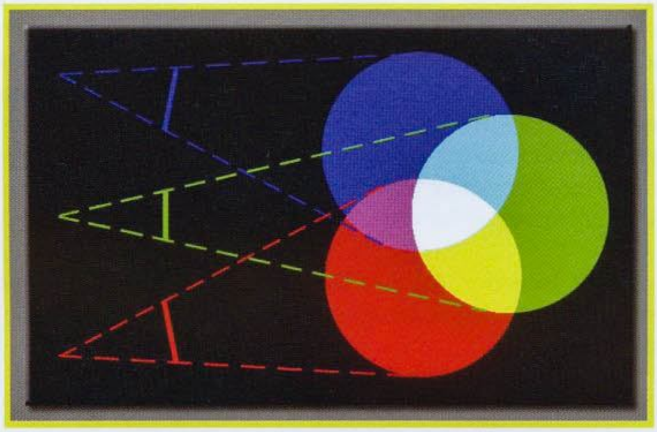
Furthermore, spectral sensibility of cones overlap, and this improves the capacity of distinguishing colors. If it were not so, humans would distinguish only three colors in the spectrum.



■ **Figure 4.62**
Absorption curves for the three cones found through experiments.

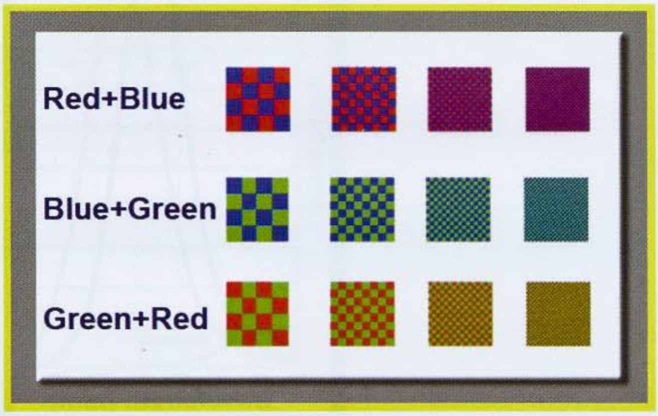
So far we have talked generically about the effects of light on human's perception system, without distinguishing between the perception of colors as result of lights coming directly from a light source, and that resulting from lights reflected by surfaces positioned between the source and the eyes.
Now this difference is to be examined in detail, starting from phenomena connected to the first situation considered above.
As seen in paragraphs dedicated the functioning of photosensitive receptors of the retina, color vision depends on the combined action of three types of cones, differently stimulated by electromagnetic waves composing light. It is possible to test that an adequate mix of different wavelength radiations produces the vision of white. This test is the opposite of decomposing white solar light into colors with a prism.
This phenomenon, that is, the fact that different wavelengths of light appear singularly colored, and together white, is called **additive mixture**. Thus the vision of the white can be considered as the perception of the sum of all radiations composing the visible spectrum.
Aiming at the creation of a reliable system able to reproduce all colors by mixing colored lights, three basic colors are used, defined primary colors. The primary colors nowadays used in TV, computer monitors and digital graphic systems, are red, green and blue.
In the following image the classic scheme of the additive synthesis is shown. It is the effect obtained by the overlapping of the three luminous rays: red, green, blue. A similar test is easy to reproduce: it is enough to use three white-light sources, each one screened with a filter of one of the three primary colors considered, and then projecting the relative rays on a neutral surface. Where the rays overlap, in the centre, white appears. Where just red and green combine, yellow appears. On the green-blue blend, cyan, a very saturated sky light blue appears. Lastly, where red and blue merge, the perceived color is magenta, a well-saturated violet red.

■ **Figure 4.63**
Classic example of spatial additive synthesis.



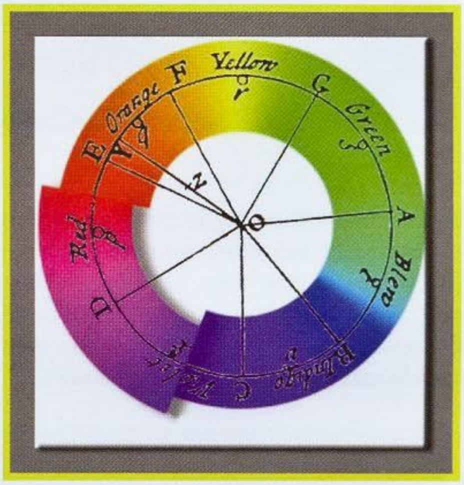
The kind of additive synthesis we have just seen is called spatial, as the effect obtained is produced by the combination of light on the same portion of space. Two other types of additive synthesis exist: the **spatial average**, and the **temporal average**.
The synthesis for spatial average takes place when different-color lights, very near to one another, are seen by the eye from a distance such that distinguishing single colors is impossible: instead of them a colored stain appears. This is the idea employed in TVs and monitors too, in which each visible point on the screen is constituted by three phosphors, photo sensible elements very near one another. One is active in red tones, one in blue tones and one in green tones. The eye reads their proximity as a unique sum stimulation, able to produce color-vision in order with the rules of the additive mixture, which is the natural mechanism of receptors present on the retina.
The next example shows an additive synthesis for spatial average: colored squares on the right are the union of dozens of semi squares red/blue, green/blue, and red/green. Retina receptors are not able to distinguish the single colors when they are so close and little, thus a unique sum-color appears, as additive mixture effect.

■ **Figure 4.64**
Example of additive synthesis for spatial average.



COLOR FEATURES

Now that the process of color sight been described biologically, it is to be explained what kind of interaction between electromagnetic waves and cones present on the retina causes the perception of different colors.
It is therefore worth starting from Newton's experiment, in 1666, when the great English physicist and mathematician discovered that white light is decomposed in colors passing through a prism.
This way Newton demonstrated that light appearing white is not monochromatic, but is the sum of several rays, each one having different wavelength. Newton also achieved the first model of color representation: a circle having in its centre white, and along the circumference, neatly arranged, the colors decomposed by the prism. The colors of the outermost parts of the visible spectrum, that is red and violet, are located in such a way to create a continuous transition. Seven colors are identified as the main ones in this model: red, orange, yellow, green, blue, indigo, violet.



■ **Figure 4.65**
Newton's color rack. The original board was white and black.

Why is white not part of the spectrum, even though it is perceived for example in the light of the sun? Other colors do not actually appear in the spectrum: black, all kind of greys, pink, lilac, brown, purple, and many others. The answer is that a spectrum-color is the perception of a monochromatic radiation with a certain wavelength, and sunlight, as all others present in nature, are a mix of monochromatic radiations.
Monochromatic radiations do not exist alone in nature. They are always a mix of monochromatic radiations with different concentrations arriving at the same time on our eyes. If a monochromatic radiation is seen alone, it is perceived as a certain color of the spectrum. But eyes are not able to perceive individual single spectrum-color within a spectrum of a non-monochromatic radiation. They perceive another color, a non-spectrum color, let us say a "global color". These are non-spectral colors. For example solar light contains all visible wavelengths, infrared and ultraviolet components as well, but the eye is not able to distinguish separate colors and perceives a global feeling of "white".
The ear works in another way: two different notes played on a piano at the same time are distinguishable, not giving a feeling resulting in an average note. The ear is analytical, while eye is synthetic. If the eye worked the same way the ear does, it would be able to see single colors of a light source corresponding to different wavelengths composing the emitted light. Instead light causes a global feeling in the eye.

Each feeling of color can be in turn decomposed into three elements, each one elementary, for it contributes to the determination of color on the part of the observer, and cannot be connected, by simplifications, to the other two elements. The elements are hue, luminosity and saturation.

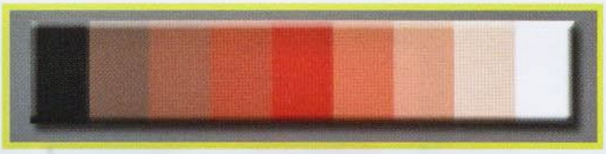
The ***hue*** is the simplest feature to understand. In common life, it is the perceiving quality that makes one give a name or another to a color. Red, green, yellow, blue are all names of hues. Physically, the quantity corresponding to hue is the wavelength: the more the light impressed on a certain point of the retina is belonging to a narrow band of wavelengths, the sharper the possibility of attributing a name to the perceived color.

■ **Figure 4.66**
Differences of hue with maximum values of saturation.

It is important to clarify that the hues the eye is able to distinguish as irreducible to other ones are the spectral colors only, that is rainbow colors, and in addition the colors deriving from combinations of spectral red and blue, the crimsons. All other colors, such as brown, salmon-pink, pink, olive-green, etc… can be defined as combinations of a certain hue with the two other features we will talk about soon. Pink, for example, is a slightly saturated red.

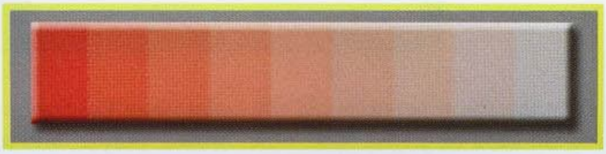
The ***luminosity*** refers to the amount of dark and bright of the color. It is determined by the degree of reflectivity of the physical surface receiving light. The higher the luminosity is, the brighter the color is.

■ **Figure 4.67**
Differences in luminosity, keeping hue and saturation constant.

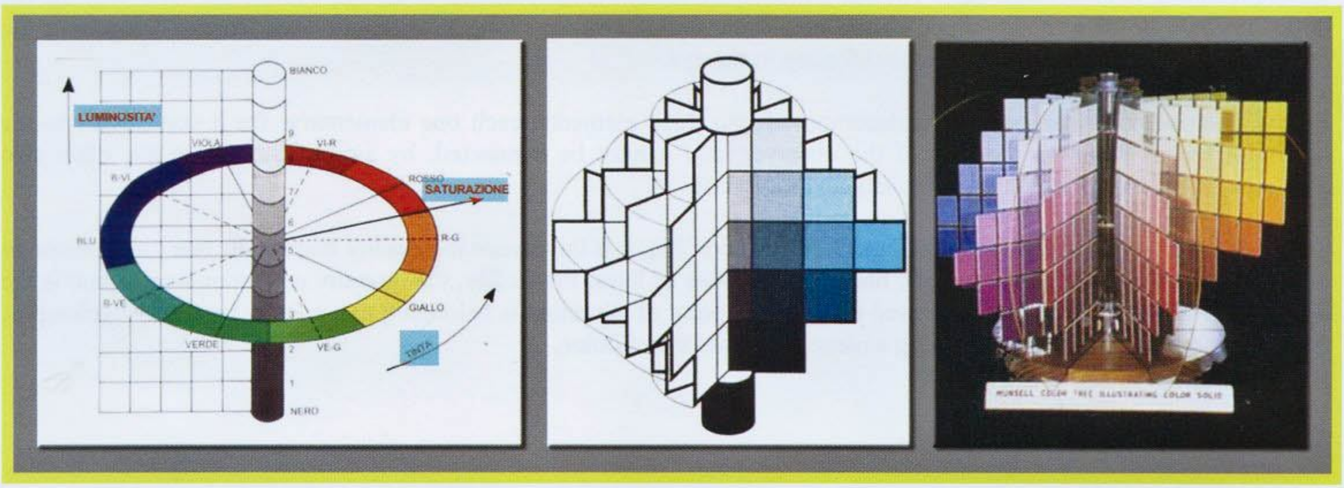


The ***saturation*** is the third element contributing to color perception. It is the measurement of purity, the intensity of the color. Spectral colors are absolutely the most saturated colors which can be perceived. They appear vivid, pure, brilliant, full, not mixed with grey in any way. On the contrary, a color with low saturation appears pale, opaque, greyish, not that recognizable as far as its hue is concerned. The reason for the difficulty in recognizing the color is that a color with low saturation derives from the mixture of lights with different wavelengths, causing a big difference with spectral colors, which are produced by very narrow band colors.

■ **Figure 4.68**
Differences in saturation, keeping hue and luminosity constant.



It is on these very three features that Albert Henry Munsell fulfilled, between 1905 and 1916, one of the most famous systems of arrangement of colors, still used today. It is considered the standard system as far as uniform perceivable displacement of colors is concerned. Munsell used the principle of perceived equidistance: the interval between each color and the following one has to be perceived as the same. Furthermore, he believed that a symmetric structure was not able to represent the relations among colors as they are perceived. In fact the classification systems already existent had used as models for their systems solids like the sphere, the semi-sphere, the pyramid, the double-cone, the octahedron, the cube, and so on. A more suitable three-dimensional structure was needed. Thus, he introduced an "opened" structure, later named "color tree", due to its external irregular profile.

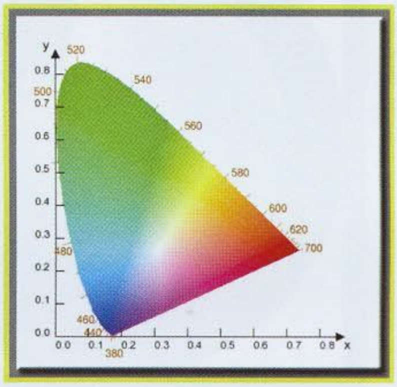


■ **Figure 4.69**
Munsell's color-classification system, its cross section, and the color tree.

COLOR SPACES

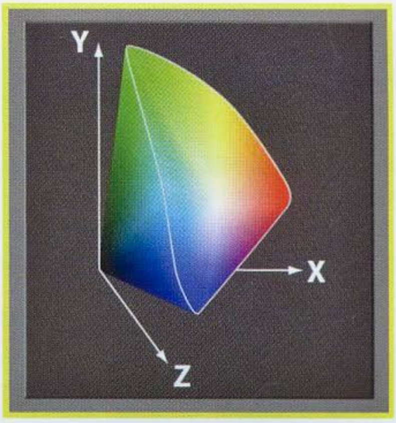
An international organization studying light and colors, the "Commission Internationale de l'Eclairage" (CIE), has developed other methods for expressing colors numerically. The first model was the color space XYZ, conceived in 1931. This chromaticity diagram does not depend on the behaviour of a particular observation or printing device, because it is based on the idea of "standard observation".

The chromaticity diagram is two-dimensional: at the centre there is white, and along the curved part of the perimeter the saturated colors of the light spectrum are placed. In anticlockwise direction: red, yellow, green, blue, violet. Central colors are not saturated, while the outermost ones are saturated. The diagram represents the hues along the perimeter with the saturations going toward the centre.



■ **Figure 4.70**
CIE XYZ diagram.

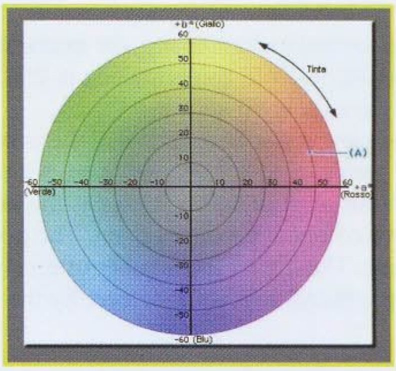
Furthermore, each color of the diagram can have a different luminosity. For example, a particular green exists also in a darker version, thus a third dimension has to be added: luminosity. If luminosity is zero, the whole diagram is black; by reducing the luminosity the centre is no longer white, but grey, and all the other colors are darker.



■ **Figure 4.71**
CIE XYZ diagram in three dimensions.

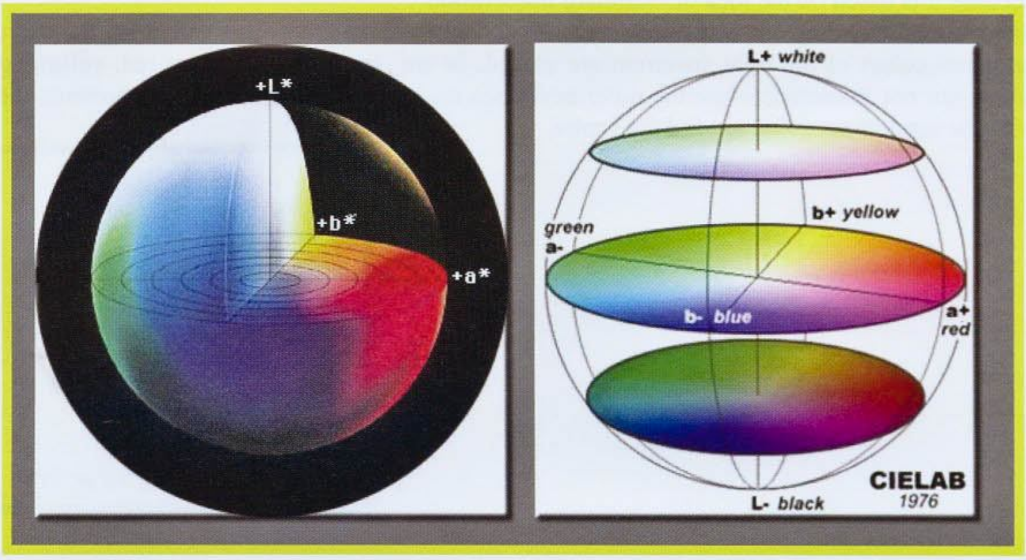
There is a problem with the CIE 1931 diagram (with coordinates x, y): the same perceptive distances do not correspond to the same geometrical distances. That is, the same differences of perceived colors do not correspond to the same distances on the chromaticity diagram. Thus, in 1976, the CIE Lab was introduced.

The Lab color space, also known as CIELAB, is nowadays the most widespread of color spaces for the measurement of the color of an object, and is widely used in many fields. In this space ***L*** means the luminosity, while ***a*** and ***b*** are the chromaticity coordinates.



■ **Figure 4.72**
Chromaticity diagram a-b.

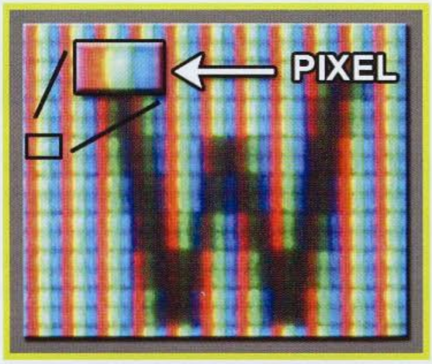
In this diagram, ***a*** and ***b*** stand for the direction of the color: *+a* is the direction of red, *-a* is the direction of green, *+ b* is the direction of yellow, and *–b* of blue. The centre is non-chromatic; when ***a*** and ***b*** increase and the point becomes far from the centre, the saturation of color increases.
There is also the ***L*** coordinate, representing the luminosity. It is perpendicular to the ***a-b*** plane, and it conventionally goes from 0 (null luminosity) to 100 (maximum luminosity, it is a white tone chosen as reference). Coordinates ***a*** and ***b*** may vary one independently from the other, but for *L*=0 and *L*=100, a and b can only be 0. Thus the Lab space as well is not only two-dimensional, but can be represented by three-dimensional geometries.



■ **Figure 4.73**
A main property of this colorimetric system is uniformity. That is, the same visual distances correspond to the same metric distances. Considering two pairs of points a-a' and b-b' within the diagram and having the same distance, it can be noticed that the difference of color perceived between a-a' and b-b' is the same.

THE COLORS AND COMPUTERS

The computer "thinks" of colors as numbers, and the screen is the instrument which allows it to represent them. This last one is an RGB device. The colors it shows are in terms of numbers, which are precisely named RGB coordinates. It works by spatial additive synthesis. Each pixel, in English **Pic**ture **El**ement, is formed by three "sub-pixels" placed side by side: a red one, a green one and a blue one which appear as a unique colored pixel if seen from afar.

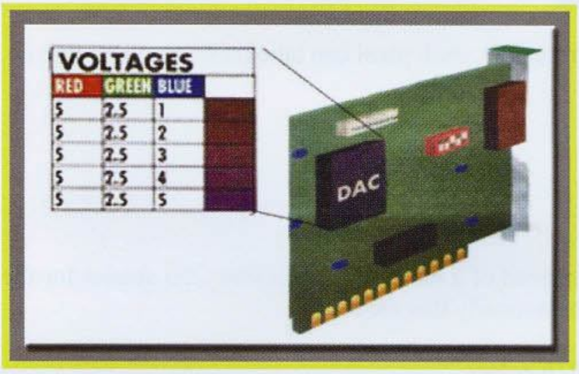


■ **Figure 4.74**
Close range image of a part of the screen. Pixels appear clear, and the RGB sub-pixels composing them also. The black tone of the letter W is given by the additive synthesis of the three sub-pixels with very low brightness.

If each red element is able to vary its luminosity, for example in 256 levels, and the same is done by also green and blue elements, the total number of possible combinations is 256 x 256 x 256 = 16,777,216 which is more than 16 million colors. One might ask why precisely 16 million.
From a perceptive point of view it has been seen that the maximum number of shades of grey that a common observer is able to distinguish in a range going from black to white is roughly 200, while the maximum number of hues distinguishable in a spectrum of pure colors between the outermost ones, violet and red, is roughly 160. Instead, the computer does not perceive colors: it produces them. The pixel is the basic unit of the screen. It could be compared, for example to a mosaic texture. The higher the number of pixels in an image, the higher its definition. Each pixel can have a specific color which is produced, in a color-screen, by the spatial additive synthesis of the three sub-pixels red, green and blue.

Within each sub-pixel some phosphors are present. These are three fluorescent substances deposited on the inner part of the screen, which have to be "stimulated" to emit light. Different levels of "stimulation" correspond to different luminosity of sub-pixels, that is, different colors. How is the level of stimulation defined?

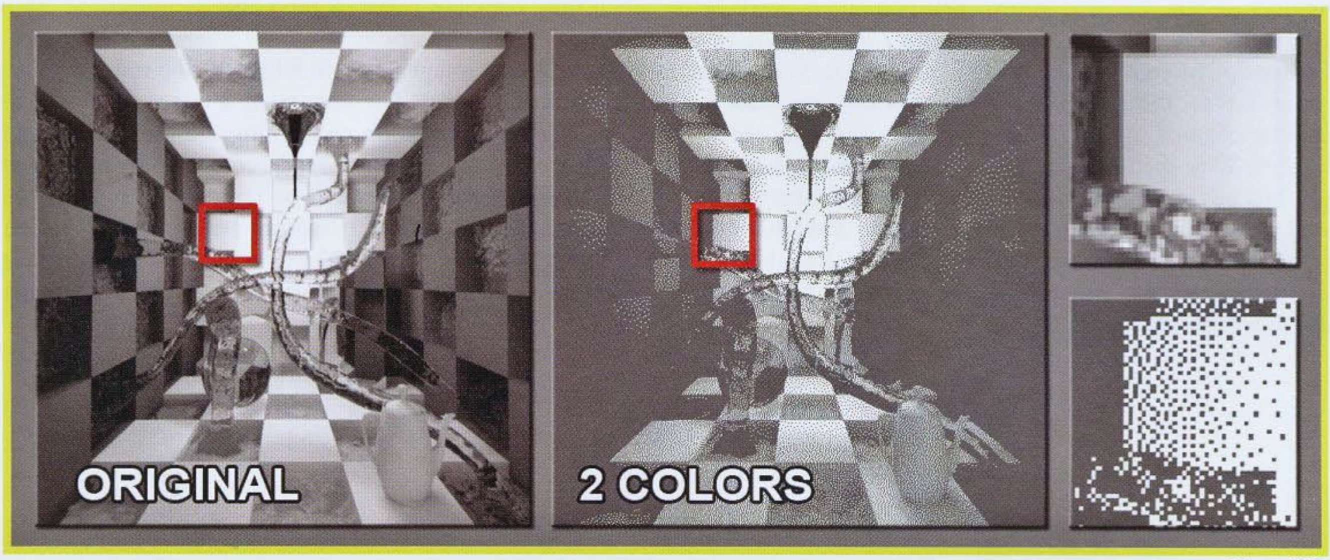
Digital signals sent by the operating system or by the applicative software are received by the video card. On the video card a device called DAC (digital-to-analogue converter) is mounted, which compares the digital values sent by the PC with a table in which the tension levels corresponding to the three primary colors needed to create the color of the pixel are contained. The table, named LUT (Look-Up Table), contains all the possible colors with their relative voltage. For example, value 1 corresponds to an output voltage of 0.1 V; value 2 to a voltage 0.2 and so on. In practice, a DAC, and therefore a LUT, is always present when a numeric information (in this case a digital input coming from the software) must deal with an analogical quantity (the tension generated by the cathode tube).



■ **Figure 4.75**
Schematic representation of a LUT table (Look-Up Table).

In modern PCs, where 256 levels for each red, green and blue sub-pixel can be obtained, the LUT contains 256x256x256 voltage values. The adaptor thus sends signals to three electronic cannons positioned in the rear part of the CRT. Each electronic cannon emits a flow of electrons; therefore there is a flow for each primary color. The intensity of each flow is controlled by the signals coming from the adaptor. Finally, electrons hit phosphors covering the inner part of the screen, which consequently become phosphorescent.

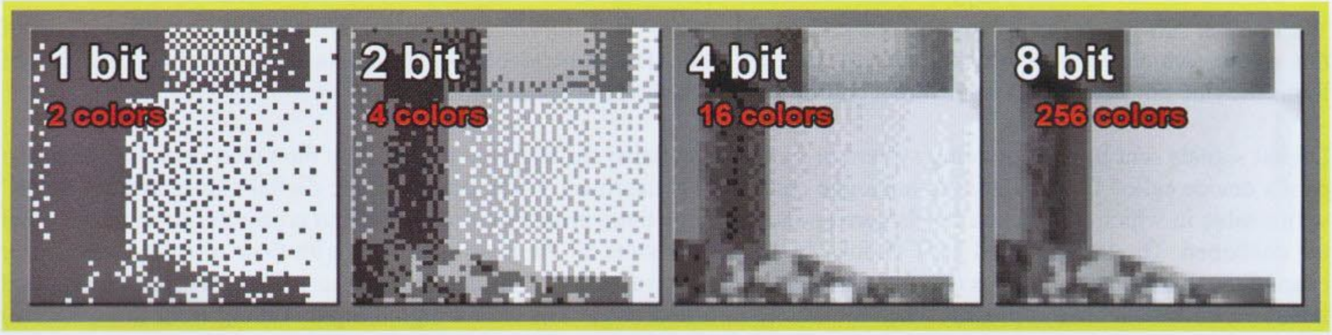
Finally, depth of the color is to be considered, which is the amount of bits needed to represent the color of a single pixel. To understand the idea of color depth, imagine an image in black and white. Imagine having to represent it using two colors only. Each pixel can be or a black or white.



■ **Figure 4.76**
Comparison between an 8-bit and a 1-bit image.

In this case we call it a 1-bit image depth .... (2^1= 2 colors),

In the case of a 2-bit image there are 4 colors .... (2^2= 4 colors).
In the case of a 4-bit image there are 16 colors .... (2^4=16 colors).
In the case of an 8-bit image there are 256 colors .... (2^8= 256 colors).



■ **Figure 4.77**
Images with different depths of color.

Using as an example a 2-bit image, each pixel can take on four colors, that is:

(0-0) = white
(0-1) = bright grey
(1-0) = dark grey
(1-1) = black

If an RGB image is used instead of a black and white one, 256 shades for the red would be available, 256 for the green and 256 for the blue. Mathematically this means:

$2^8 \text{ x } 2^8 \text{ x } 2^8 = 2^{24} = 16{,}777{,}216$ colors.

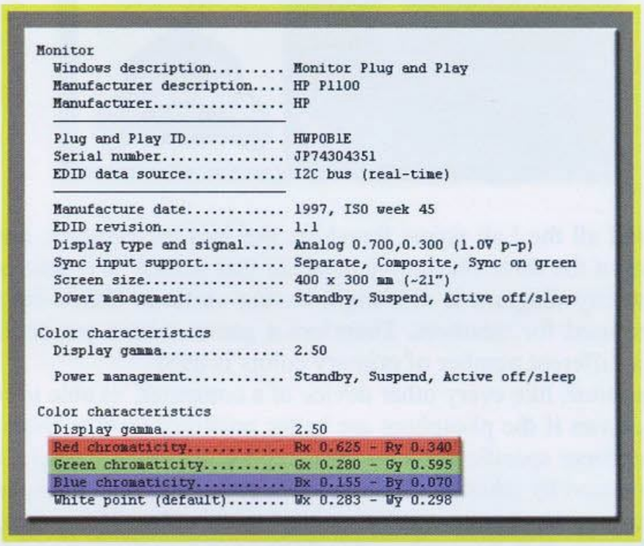
In other words, we call this 24-bit image depth. Each pixel can take on one of the 16,777,216 colors available. The human eye is able to perceive 200 gradations and roughly 1 million colors at the most. It appears clear that 16M colors are in fact too many for the human eye. Thus, one might consider using fewer colors. In fact, in a certain period PCs worked with High Color, or 16 bits, and not in Truecolor or 24 bits.

In the case of a 16-bit image 65,536 colors were available……($2^{16} = 65{,}536$ colors).

A 16-bit depth is usually divided into 5 red and blue bits, and 6 green bits (green has more bits because the eye is more sensitive to this color). These colors are in fact too few, and do not reach a million. It was then possible to increase the number of bits, but it is known that computers work better when they compute bytes (8 bits) and its multiples. As a consequence 8, 16 or 24 –bits images are to be preferred.
Perhaps often you have heard the term 32 bit. Well, these images are 24-bit images, that is 8 bits for the red (256 red-shades), 8 for the green (256 green-shades), and 8 for the blue (256 blue-shades). The remaining 8 bits are devoted to another channel, named alpha (256 grey-shades), usually used for storing information concerning the transparency of images. With the last generation image formats, such as the HDRI, it is possible even to reach color depths of 96 bits (32 bits per channel)! In this case, in comparison to "traditional" images where colors can be represented in a ranges varying between [0-255] for 8-bit images, and [0-16M] for 24-bit images, $4.3\text{x}10^9 = 2^{32}$ colors per channel might be reached, that is 4,3000,000,000. For representing this high amount of data floating point values raised to a certain power of 10 are used. An example of color in HDRI format can be described with simple integers [0-255], but with decimals too. For example $5{,}467\text{x}10^3$.

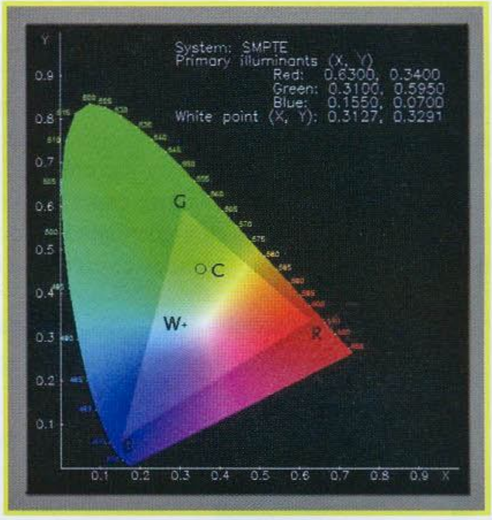
THE SCREEN AND ITS FEATURES

The first feature in a display is the group of chromatic coordinates of its three phosphors: the x, y coordinates respectively belonging to the red phosphor, the green one, and the blue one relatively to the CIE diagram. For example the phosphors of a SONY Trinitron CRT have these chromatic coordinates x and y:



■ **Figure 4.78**
For the analysis of the screen a freeware program has been used: MonInfo - Monitor Asset Manager.

All the colors which can be generated by a monitor are precisely those which, within the CIE diagram, are contained inside a triangle having as its summit the colors of the three phosphors of the screen. This triangle is the gamut of the monitor. Thus, with apposite mathematical formulas, one can calculate the CIE coordinates (X, Y, Z) of any color generated by the monitor, and therefore contained inside its gamut. The gamut is the range of colors a device is able to produce, reproduces or catches, and usually is a subset of visible colors.



■ **Figure 4.79**
Different models exist for describing colors, such as the RGB, CMYK, etc... The term gamut of a color-model means all the colors which can be described by that specific model. When a model is not able to produce a color, it is said that the color is "out of range" relatively to that gamut.

In principle, it is very simple to construct the gamut of a screen. One starts from a table containing all the possible RGB combinations which are 256 x 256 x 256, that is 16,777,216. Then the absolute Lab coordinates of the color produced by the monitor with that specific RGB combination is written for each combination. To carry out this operation a colorimeter is needed, which reads the color produced by the monitor by a specific term RGB with special optical sensors.

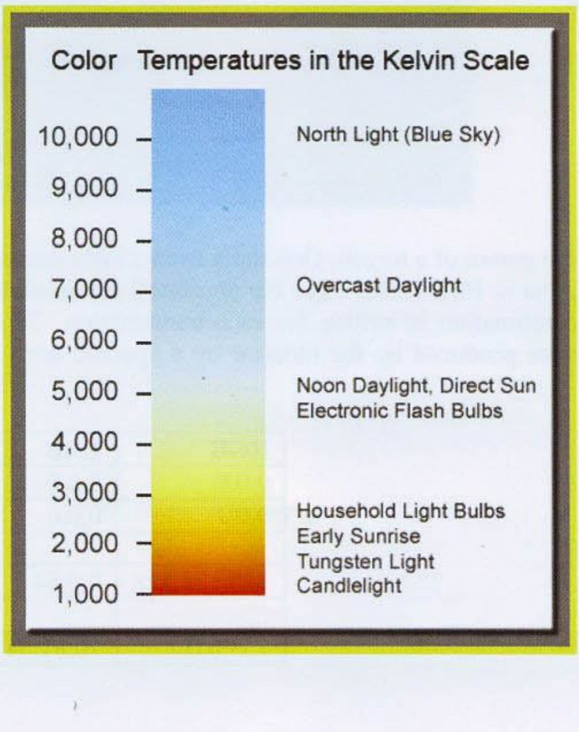
RGB	LAB
0,0,0	0,0,0
0,0,1	0,0,0
…	
128,128,128	0,0,54
….	
255,0,0	70,81,54

■ **Figure 4.80**
Fairly good colorimeters, such as the Spider of Colorvision, are able to give excellent performances costing just a few hundred Euros. For professional graphics it is an essential tool.

If this table is achieved and all the Lab points found are put into the diagram, a triangular distribution appears, where the vertexes are the colors of the three phosphors used for that screen. It is also possible to notice that a gamut whose projection on the chromaticity diagram is a triangle having vertexes linked by straight lines, is a typical feature of additive mixing of colors, used for monitors. Therefore a gamut with a projection which is triangular, quadrangular, hexagonal, etc… means a different number of primary colors is used.
It appears clear that the monitor, like every other device of a computer, is able to reproduce just a part of the colors that the human eye perceives. Even if the phosphors are better positioned, there would always be non-reproducible colors. Thus, each screen has its proper specific colors gamut, represented by a triangle, or, as it is said, by its color-space of the RGB-type (that is, produced by additive mixing of red, green and blue phosphors). But saying that an RGB color is [120, 20, 230] is in any case an inaccurate way of specifying the color. These three numbers lead to different results on monitors which have different phosphors. Thus the RGB is a non-absolute color space: a device-dependent system, depending on the specific screen used. Instead, the CIE spaces allow one to specify the color in an absolute way, independently on the device.

The second feature defined by the display is the white-point. This is an important parameter which can be changed by the user, contrarily to what happens for the chromatic coordinates of phosphors. The first step for the comprehension of the white-point is the definition of color-temperature.
The color-temperature of a light source is the numerical measurement of its chromatic appearance. It is based on the principle by which each object, heated up to a high enough temperature, emits light, and the color of this light changes predictably when the temperature increases. The system is based on the changes of color in a body, heated and brought from a black cold condition to an incandescent white condition. As the temperature increases, the black body gradually passes from red to orange, yellow, white and finally to bluish white. The color-temperature of a light source is exactly that temperature, expressed in Kelvin degrees, whereby the color of the source is exactly black. For many sources it is not possible to obtain a perfect correspondence. In those cases, one refers to the nearest correspondence possible to the perfect one, and the color is described by means of a correlated color-temperature. For example, a fluorescent tube with color-temperature of 4000 K has a chromatic appearance similar to a black body heated up to 4000K.

■ **Figure 4.81**
Table of temperature/color equivalence, using absolute degrees (Kelvin degrees). To convert to Celsius degrees one has to subtract 273 from the written number. Ex. 2,000 Kelvin degrees correspond to 1,727 Celsius degrees.

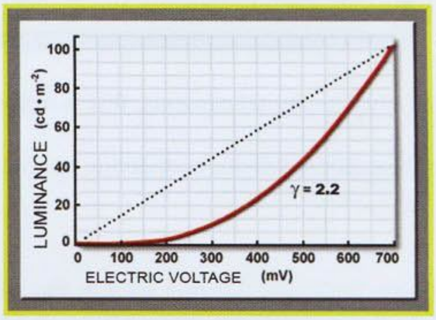


The white point, instead, defines the screen's white color. At first sight, having to define the white may seem strange. But the point is, as seen before in defining color-temperature, a certain white might be colder or hotter according to the temperature it has. Furthermore, white color-temperature depends on the device used as well. For example, as far as paper printing is concerned, white is defined at 5000 K, because standard light used for evaluating the printout typically has this value. In this case the color-temperature of the lamp is measured, as paper is not able to emit light on its own. As for monitors, a white color-temperature set to 6500 K it is strongly advisable, if the aim is to produce photographic printouts. This value is the nearest one to neutral white for most monitors. The setting is usually included in setup functions for modern screens. It would be wrong to set the temperature to 5000 K, even though this might seem logic for it is the same temperature of the paper on which one prints. In fact a temperature of 5000 K would make it rather yellowish, and, in any case, the visualized image would not correspond to the printouts which are to be produced, even though they would be seen under a light at 5000 K. This difference derives from the different physical medium. Paper and screen clearly have a different way of representing the same image. This is an interesting point: monitors are delivered with a pre-selected color-temperature, with factory settings, around 9000/9300 K. Take note: if one gets used to a screen set to 9000 K (the factory value), by lowering the color temperature to 6.500 K, everything will seem yellow. This is normal, and depends on human brain, which has to get used to the new white. Once one has got over this initial feeling, after a few days use, one feels no difference anymore: the brain has got used to the new white, in favour of printing standards.

Lastly, the third parameter, the long-awaited gamma factor. Technically, the intensity of light emitted by each phosphor on the screen in a CRT monitor, called Luminance L, does not linearly depend on the voltage V controlling the electron beams coming from the RGB cannons which excite phosphors, but follows a non-linear trend, set by the formula:

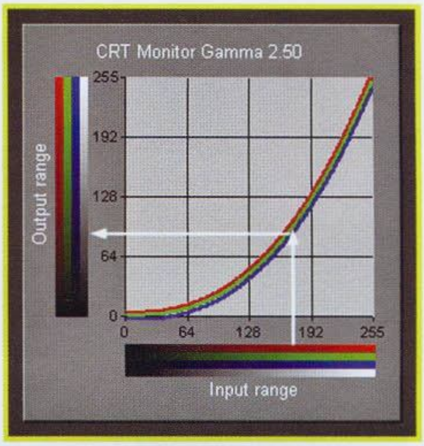
$$L = V^g$$

where the letter g represents the gamma factor γ of the monitor. In the '30s it would have been too expensive to insert an analogical circuit in a television to make this relation become linear. Thus it has been preferred not to install any correction system.



■ **Figure 4.82**
Chart showing the non-linear relation in cathode tube.

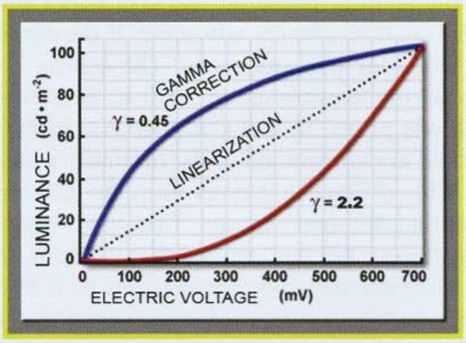
In other words, the gamma factor represents the relation standing between the luminosity of pixels as they appear on the monitor and their real RGB value. It is known that the maximum intensity the red phosphor can assume is the RGB value [255, 0, 0]. And probably some might think that the RGB = [128, 0, 0] appears with half the luminosity of the RGB = [255, 0, 0], and the RGB = [64, 0, 0] appears as the fourth part. Unfortunately it is not so: the gamma is to blame!



■ **Figure 4.83**
Non-linear relation between input RGB values and output (monitor) values.

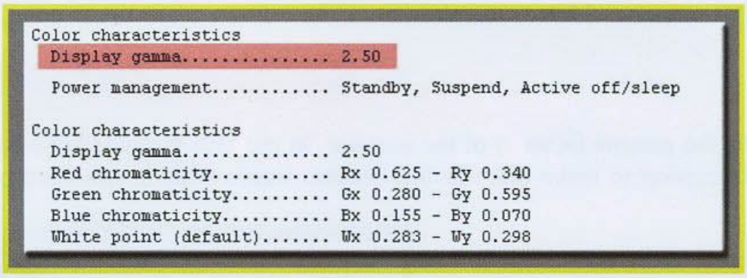
As one might notice, there is a strong flattening for dark hues, meaning non-linearity of monitors. The black tones are made even blacker, and there is loss of detail in areas in shadow if no correction is carried out.

As far as TV transmissions are concerned, to solve this problem, it has been preferable to introduce a distortion directly on the video before its spreading on the air. The process of pre-compensation for this non-linearity is known as Gamma Correction. The required function is approximately a power law of 1/g, that is the inverse of gamma (1 / 2.2 = 0.45). The result is a complete linearization of the image on the CRT.



■ **Figure 4.84**
Gamma Correction applies an inverse function so as to linearize monitor's gamma factor.

In a monitor the gamma factor cannot be directly changed from the screen, like the white point. It can be changed by modifying the LUT of the video card, by applying a gamma correction in order to apply the desired gamma factor to the monitor. Thanks to the software Moninfo it is also possible to know which gamma the monitor has: 2.50.



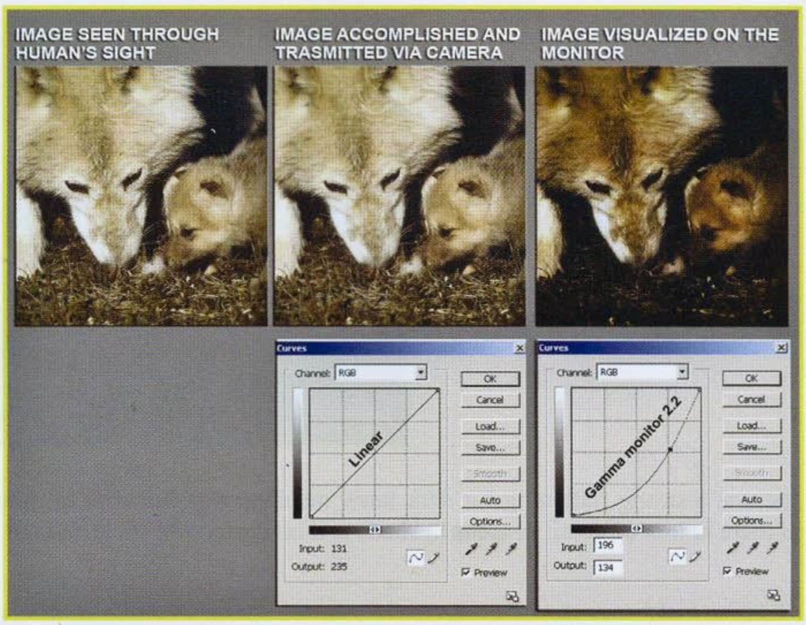
■ **Figure 4.85**
Monitor features which can be seen with the freeware MonInfo program.

In the case of CCD devices, such as digital cameras, this is different.

Digital cameras also have their gamma factor connecting the output tension V to the irradiance E:

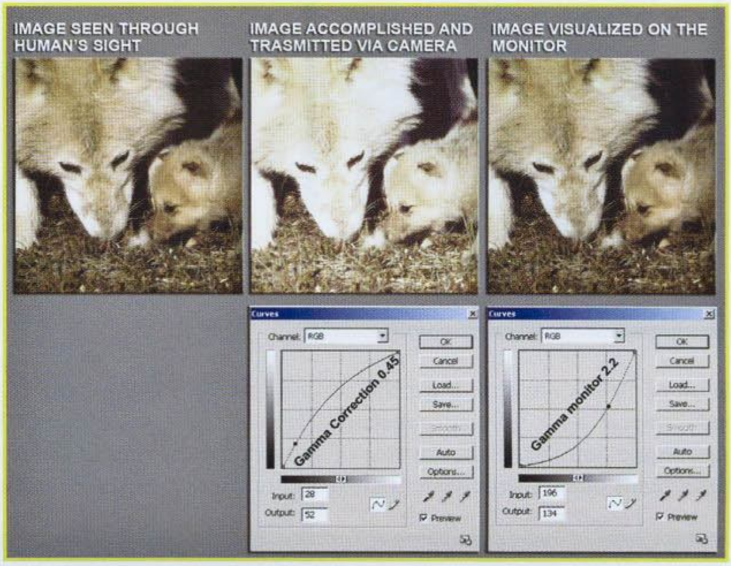
$V = E^{1/g}$

In this case the gamma factor is usually set to 1 or 2.2. On technical data one prefers to write down the power of E directly, that is 1/g, which becomes 1 (in case 1/g = 1/1), or 0.45 (in case 1/g = 1/2.2).
The reaction of the CCD sensor of the digicam is linear with irradiance, thus the physical g of a camera is 1. but connecting a digital camera with g = 1 to a monitor for PC, the user would see the scene less contrasted and flatted on darks for medium-low values of irradiance, and much brighter and contrasted for higher irradiances. The luminance of the scene would be reproduced by the monitor in a distorted way compared to reality.



■ **Figure 4.86**
The image transmitted by the camera is linear. Due to the gamma factor of the screen, the image appears much darker.

For this reason many digital cameras have an option allowing one to change the gamma factor. If g = 2.2 is chosen for the digital camera, monitor luminance is roughly linear with the irradiance of the scene, which is to be then reproduced in a realistic way. That means the image transmitted to the camera has a correction which is roughly the inverse of monitor gamma factor. The two gamma factors are the inverse of each other, and the final result is a linear image on the monitor.



■ **Figure 4.87**
The image seen by the naked eye is linear. The camera acquires in a linear way, but transmits in a non-linear way. This phenomenon has been reproduced with curves in Photoshop.

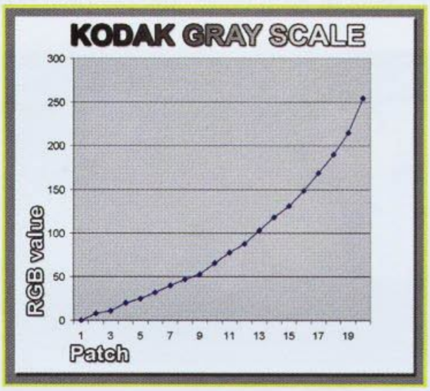
The knowledge of gamma factors of digital cameras used in shooting images is essential, provided that the user knows how to use it in the best way, without making interpretation mistakes or errors during the reading of data.
While for professional cameras and industrial cameras handbooks always define this parameter explicitly, for commercial cameras the gamma factor is not even mentioned or written in the technical data. To know its value one has to measure it. Several measurements on gamma factors are not specified in technical data, and carried out in laboratories, have measured values of 2.5. Images on the CRT monitor in this case are pretty linear compared to real luminance.
One has to pay attention to images coming from cameras supporting the RAW format, which allows one to memorize raw data coming from the sensor of the camera. The RAW format typically consists of 10 or 12 bits for each pixel, and contains all the information concerning the image shot, without white balancing, color compensation, contrast increase and gamma correction. The image is thus stored in a linear way. This format is preferable to .jpg or .tif formats in some cases (as happens instead for RAW images), as these last ones are the result of an invisible modification carried out via software by the camera and not by the user, precisely like the gamma correction (other than all the other corrections such as white balancing, etc…). In this case the user decides how to correct the image. Furthermore, the RAW format allows one to generate 12-bit images, that is, it has a higher chromatic dynamic compared to .jpg and .tif formats.

Another important facet involves the number 2.2. If one observes Kodak's table of greys, it may be noticed that they are distributed in an uniform way, from the brightest to the darkest.

■ **Figure 4.88**
Kodak's table for control and setting of grey tones.

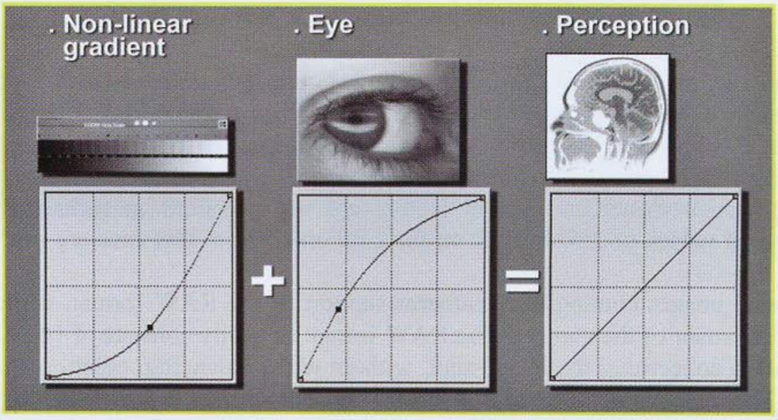
Visually, they appear to lose luminosity gradually, or, in other words, the perceived difference between the luminosity of two consecutive greys is constant. Instead, the luminous intensity reflected by several rectangles appears to decrease not linearly.

■ **Figure 4.89**
Kodak's greys-scale non-linearity.

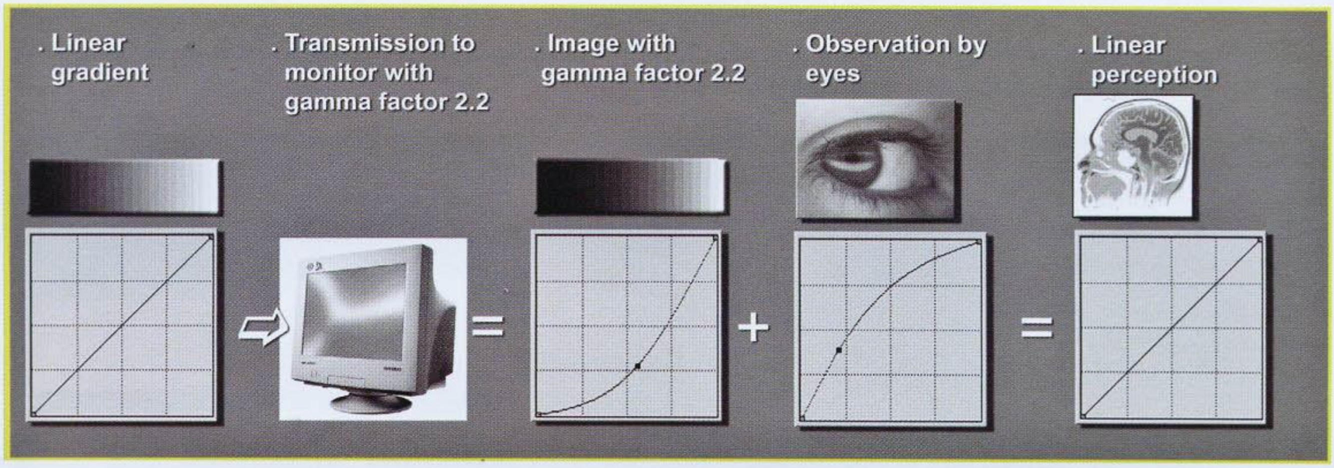


For the first rectangle, a value of 100 is measured, for the second one 90, for the third 75, for the fifth 50, and so on. Why instead does luminosity appear as a gradual constant? Because the eye does not have a linear perception of luminance. Instead, it is logarithmic, similar to the inverse of the gamma factor 2.2 of the monitor. As a consequence, observing a non linear gradient leads one to perceive a gradual and linear decrease of luminosity.

■ **Figure 4.90**
Linear perception of a non-linear gradient caused by the logarithmic vision of the human eye.

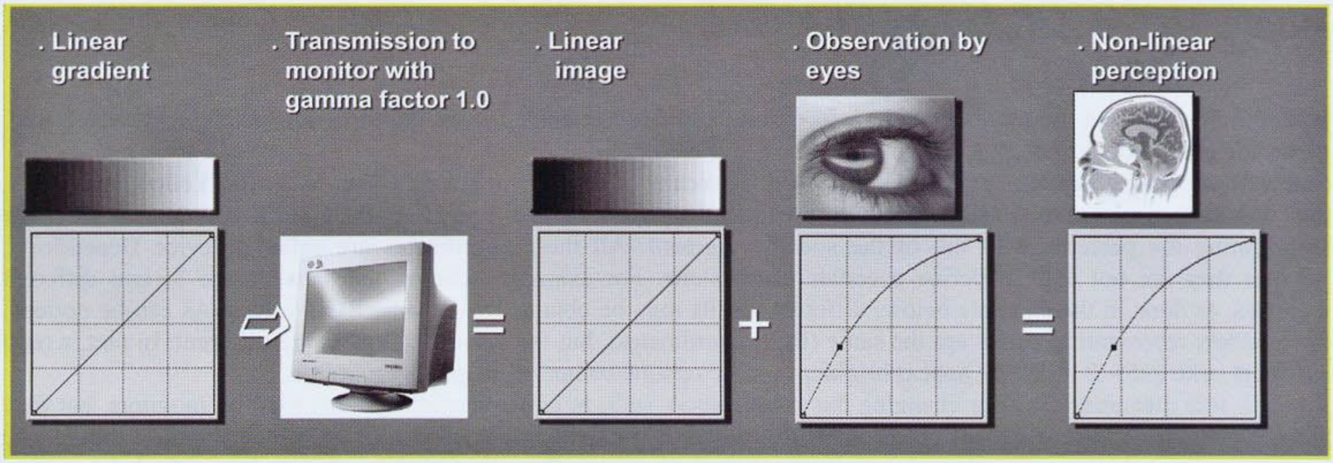


As a consequence one can make the setting by calibrating the monitor with a gamma factor of 2.2, a gradient from black to white with RGB values growing linearly appears to visually grow uniformly when seen on the monitor.



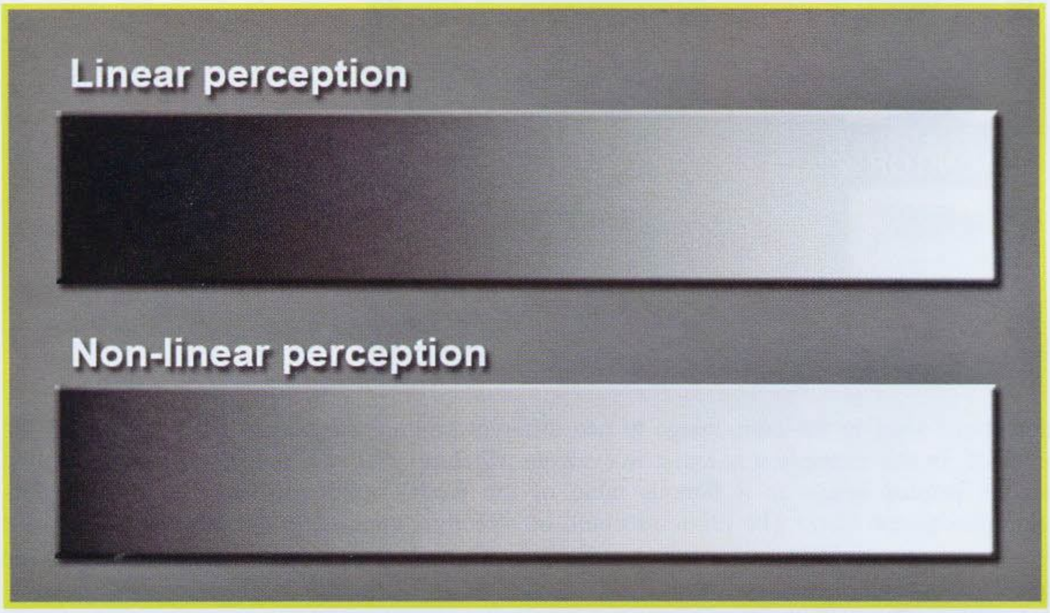
■ **Figure 4.91**
Relation between linear image and visual perception thanks to the CRT system 2.2.

The monitor could also be calibrated with gamma = 1, making its behaviour similar to an electronic sensor (digital cameras and scanners), in order to make the correspondence between RGB values and the luminance linear.



■ **Figure 4.92**
Relationship between the linear image and the visual perception with a CRT system 1.0.

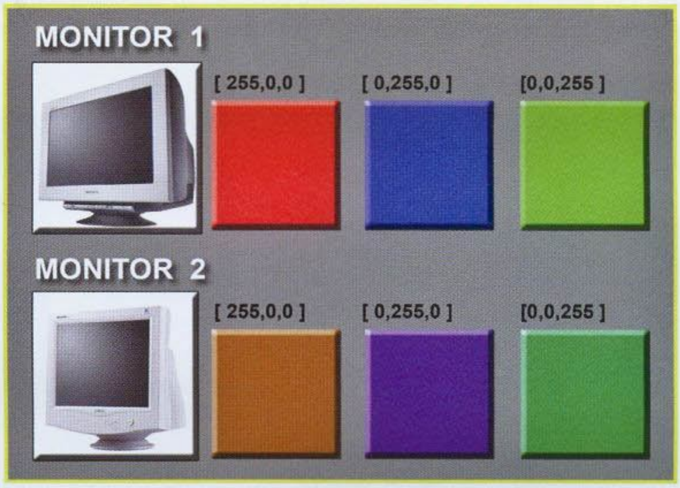
In this last case the luminosity variation is visually much sharper than the previous one. It would be much more uncomfortable to work using RGB. For example, it would be more difficult to choose a dark grey than a bright grey, the non-linear gradient "compresses" the low luminosity range of colors into a more narrow space compared to the bright components.



■ **Figure 4.93**
Linear and non-linear gradients.

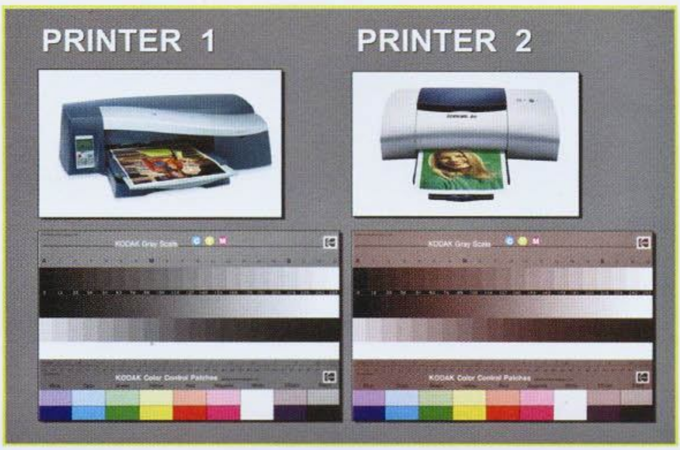
It should be clear that the monitor works with spatial additive synthesis, where the three sub-pixels, respectively red, green and blue, produce colors if seen from afar.
Similarly, a printing machine generating all colors by means of the four pigments Cyan, Magenta, Yellow and Black, uses coordinates related to the amount of ink spread on the paper. These are precisely the CMYK coordinates, each one representing the percentage of ink of the specific pigment. All these coordinates are called "Device Dependent", because they are not able to define a certain color alone. For example, sending the same image to two different monitors, as done in the example below, different results may be obtained with same RGB values. As can be noticed, the numbers representing colors are the same (RGB coordinates), but the visualized colors are different. In fact, a pure red RGB color [255 ,0 ,0] might appear brighter or more opaque compared to another monitor.
It is also true that commands for changing the visualization, such as luminosity or contrast, exist on monitors, but they are not enough to have a decent correspondence. Another problem might arise too: on which monitor should one operate to obtain this correspondence? Just on one or on both of them? One could choose on the basis of what one likes more at first "sight"; in this case monitor 1, because it shows more saturated colors. Even though we would obtain a good correspondence, how would we know that the rest of the world has calibrated the monitor in the same way, and therefore sees images as we do?

■ **Figure 4.94**
The same RGB coordinates seen on different monitors never appear the same.



The same thing happens for prints: sending the same image to two different printing machines CMYK almost all the time different prints are obtained. In this example it is easier to evaluate which printing machine works better, the first one in this case, because the printed image is a famous table of the Kodak used appositely to make colors correspondence easier during photograph steps. The printouts could be directly compared to the original Kodak card which has just been purchased and one could find convenient "adjustments" in all images before sending them to the printer. Maybe using curves, levels and all Photoshop's tools.

■ **Figure 4.95**
The same image, printed with two different devices, most of the time does not correspond to what is seen on the screen. Furthermore, two different printing machines always tend to create different prints. In this case the second printer is affected by a strong dominance of red.



The method we have just described is not the best one in any case. Instead, it is to be avoided. The management of the color must not be done manually by correcting the images with curves and levels before sending them to the device. Moreover, it should be carried out not by the user, manually, but by means of a software which does this automatically and precisely this operation by using the information contained in the profiles.

Two basic ideas have been outlined until now:

1. the RGB and CMYK coordinates are not absolute coordinates, and therefore they cannot describe colors.
2. For defining absolute coordinates a model must be built before: the CIELab model has been chosen here.

An ICC profile creates a correspondence between coordinates concerning the visualization tool and the absolute coordinates of the CIELab model. The International Color Consortium ICC is an independent organization founded in 1993 to develop and maintain an open standard, for operating systems and multiplatforms for digital management of the color. For this purpose, it issues its specifics of building mode and employment of color profiles. A color profile complying to these specifics is called ICC profile. The eight foundation members are: Adobe, Agfa, Apple, Kodak, Taligent, Microsoft, Sun and Silicon Graphics.
Let us consider, for example, the case of a monitor. In the ICC profile associated to the screen a table relating RGB and Lab coordinates is present.

RGB	LAB
0 , 0 , 0	0 , 0 , 0
0 , 0 , 1	0 , 0 , 0
…	…
0 , 0 , 10	0.05 , 0.26 , -1
…	…
24 , 60 , 213	33 , 33.72 , -86.26
….	…
255 , 255 , 255	100 , 0 , 0

■ **Figure 4.96**
Table of correspondences between relative and absolute values in a screen.

To create this table one starts by sending the monitor the RGB = [0, 0, 0] coordinates. With a colorimeter the absolute Lab coordinates of the visualized color are measured, and written in the first row. Then one goes on with the coordinates RGB = [0 ,0, 1], writing the coordinates in the second row, and so on. In such a way one creates a complete description of the screen. If the same process is carried out for another monitor other Lab values are to be found. Earlier we stated that roughly 16 million colors can be obtained by 24-bit RGB images. One should therefore measure 16 million combinations and create a table containing the same number of rows. Obviously, for practical reasons, fewer measurements are carried out, in a distributed way. It is also clear that the higher the number of measurements carried out is, the better the profile is, while the missing rows are to be calculated by interpolation.
Now the profiles may be used for having a color correspondence between monitors. For example, if the color relative to the coordinates RGB = [24, 60, 213] is observed on the first monitor, it is simple for us to find in the nearby chart the correspondent RGB color on the second monitor, which is RGB = [44, 40, 183]. In this way it is possible to see the same color on both the screens.

MONITOR 1

RGB	LAB
0 , 0 , 0	0 , 0 , 0
0 , 0 , 1	0 , 0 , 0
…	…
0 , 0 , 10	0.05 , 0.26 , -1
…	…
24 , 60 , 213	33 , 33.72 , -86.26
…	…
44 , 40 , 183	26.81 , 38.45 , -76
….	…
255 , 255 , 255	100 , 0 , 0

MONITOR 2

RGB	LAB
0 , 0 , 0	0 , 0 , 0
0 , 0 , 1	0 , 0 , -0.06
…	…
0 , 0 , 10	0.19 , 0.89 , -4.74
…	…
24 , 60 , 213	38 , 24.97 , -92.37
…	…
44 , 40 , 183	33 , 33.72 , -86.26
….	…
255 , 255 , 255	100 , 0 , 0

■ **Figure 4.97**
Example of conversion between two monitors.

Converting an image from one profile to another means repeating the procedure we have just described for all the pixels in the image. In other words carrying out a conversion between RGB profiles leads to changing the RGB coordinates so as to keep the Lab ones the same. The same goes for the CMYK coordinates. The charts below show how to carry out a conversion from RGB to CMYK: for example if one wants to reproduce the color that the monitor visualizes with the values RGB = [1, 73, 166], the values [100, 70, 0, 0] have to be sent to the CMYK printer.

■ **Figure 4.98**
Example of conversion between a monitor and a printing machine.

MONITOR RGB

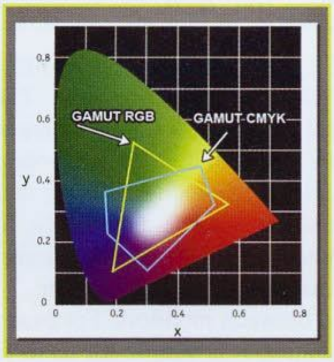
RGB	LAB
0 , 0 , 0	0 , 0 , 0
0 , 0 , 1	0 , 0 , 0
…	…
0 , 0 , 255	31.5 , 60.1 , -107.0
…	…
1 , 73 , 166	30.7 , 14.4 , -68.9
…	…
44 , 40 , 183	26.81 , 38.45 , -76
….	…
255 , 255 , 255	99.6 , 0 , 0

STAMPANTE CMYK

CMYK	LAB
0 , 0 , 0 , 0	0 , 0 ,0
0 , 0 , 0 , 1	0 , 0 , -0.06
…	…
100 , 86.4 , 0 , 0	0.19 , 0.89 , -4.74
…	…
93.2 , 65.5 , 0 , 0	38 , 24.97 , -92.37
…	…
100 , 70 , 0 , 0	30.7 , 14.4 , -68.9
….	…
87 , 81 , 70.3 , 66	13.5 , -0.01 , -1.22

These charts show a problem we have not yet considered. If one wants to convert the blue tone which the monitor visualizes with the coordinates RGB = [0, 0, 255], which CMYK coordinates have to be sent to the printing machine? Following the same procedure seen before a bitter surprise is in store for us: in the printing machine profile chart the searched Lab coordinates are not found, because they are "out of gamut"!
As we have already said, all the colors a device is able to reproduce is called Gamut, and each device has its own Gamma. In this case the color we want to print is not reproducible because it is beyond the printer's gamut.

■ **Figure 4.99**
Graphical representation of the RGB and CMYK gamut.

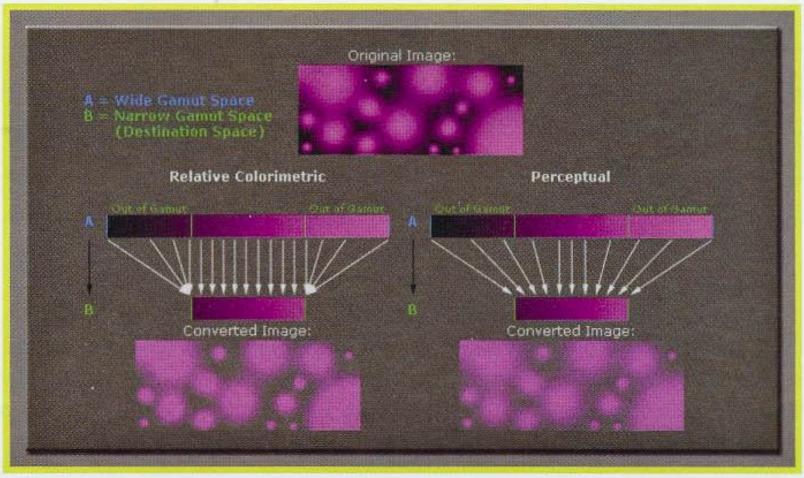


Therefore, some approximations must be made when one wants to convert colors from one device to another, or from one profile to another. Different approximation methods exist. There is not a best one among them. For this reason the ICC profiles offer four different systems called rendering intents. Without going too far into this topic, four rendering intents are defined here:

. Absolute colorimetric intent;
. Relative colorimetric intent;
. Perceptive intent;
. Saturation intent.

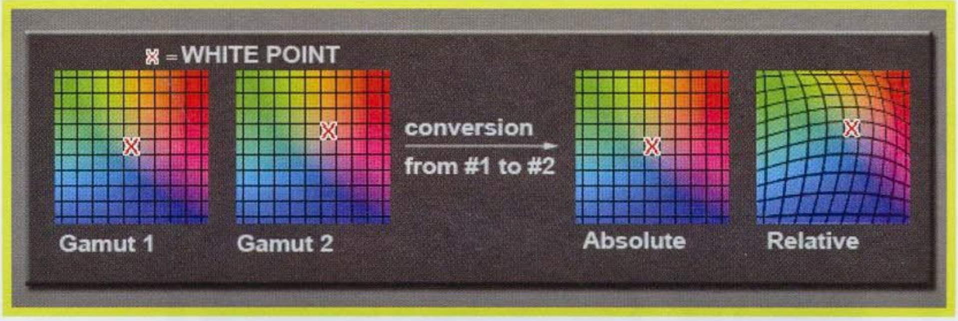
The colorimetric, relative and perceptive intents are probably the most used in digital photography.

■ **Figure 4.100**
Different rendering intents.



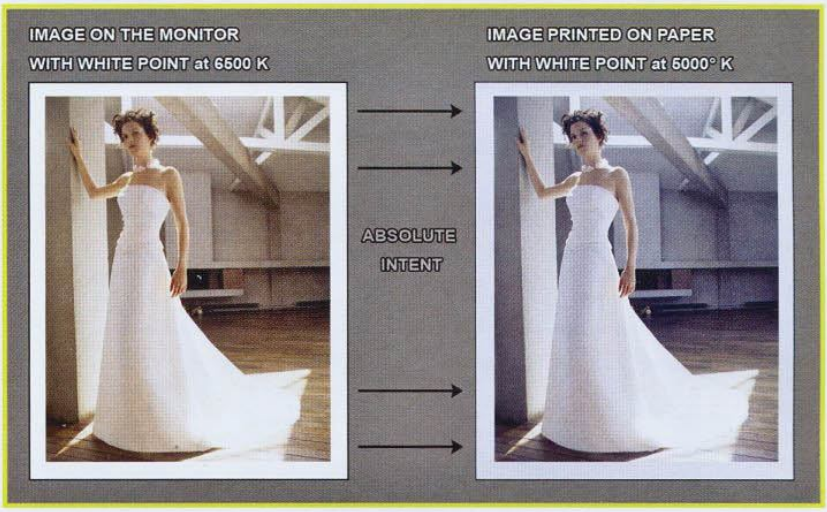
Essentially the two methods are different in how they deal with out-of-gamut colors. The first one, that is the relative colorimetric intent, cuts off the out-of-gamut colors, approximating them to the colors closer to the perimeter. The perceptive one, instead, carries out a uniform approximation on the whole the range of colors, even those which have a real correspondence between two gamuts.
It can be noticed that the perceptive method maintains a generic uniformity in the entire image. Whereas by using the relative method some areas result much more distinct, like the centre of the spots, where a strong flattening of the colors takes place. Therefore, in practice, it is worth using the relative intent if the image to be printed is completely contained in the destination gamma, otherwise, if the gamut of the image is much wider than the destination, it is advisable to use the perceptive intent.

As far as the difference between the absolute and relative colorimetric method is concerned, they are very similar systems. The only difference concerns the white point. The white point represents the position of the most pure and luminous color within the gamut. For the relative method the original white point coincides with that of destination. To understand this difference look the two theoretical gamuts, perfectly square and equal. The only difference is the position of the white point, marked with an x.



■ **Figure 4.101**
Difference between the relative and the absolute intent.

The absolute method preserves the white point, while the relative one moves it in such a way to keep the old white point aligned with the new one. This is the reason why relative intent is the most used in photography. If the white point of the image depends on user's choice, that of destination usually corresponds to the one of paper used for the printing.
Consider a white sheet of paper, observed under the sun light. Its color-temperature is roughly 5000 K. Let us suppose that the monitor is calibrated to 6500 K, a very common situation. The concept of grey is relative to what is considered white, thus details appearing neutral on the monitor seem more bluish if compared with the paper's white (or, vice versa, the paper's white appears to be more yellowish if compared to monitor white), because it has a "hotter" white point. The monitor white itself, once converted with this intent, would appear as a cooler color compared to paper. Therefore the printing machine should use a little bit of cyan and magenta where the pure white of the paper should instead be maintained. The result would be that white areas on the monitor, which would have to be preserved on the paper, would instead be colored with a cold hue.



■ **Figure 4.102**
Effect of the absolute rendering intent on the image's white point.

The original picture is quite neutral, even though it is to be noticed that the wedding dress is not white, but ivory as it was in reality. In the printing simulation it is clear that the image is cooled, as expected: the dress has become almost white compared to the outermost edges; the inside of the flat has taken on a bluish unnatural color, the general rendering is definitely colder, and the originally white areas, such as the internal edges of the picture, are "dirtied" by the ink of the printer which has to simulate the monitor's white, that is the soft bluish color due to the temperature of 6500 K.
Saturation intent is the least interesting from the photographic point of view, and therefore it is not going to be explained.

MONITOR CALIBRATION

The calibration process is composed by two main operations: real calibration and profiling.
Calibration is the operation allowing one to set the luminosity, the contrast and the color balance of the monitor in the most accurate way, by means of controls of the monitor and several software and/or hardware tools. These operations change the physical behaviour of the monitor: they are not just a simple software compensation. In practice, calibrating a device means bringing it to a known and stable state.

Profiling is the operation allowing one to memorize the exact profile of the calibrated monitor: in other words, it is the memorization of a chart of numerical values which, once passed on to the video card, guarantees that the colors shown on the monitor are accurate. The values in this table "explain" to the video card which signal the monitor needs to reproduce a certain color accurately. This profile is typical and unique for every monitor, a sort of fingerprint. For this reason doing the calibration and the profiling for each monitor is very important. It is almost impossible to have two identical monitors, for model and brand, with the same profile. Furthermore, these operations must be carried out very often, at least once a month. Electronic components are affected by a loosening of calibration, slow but constant. At the end of this process an ICC profile is generated and used in graphic programs guaranteeing a correct color management.

To better understand the calibration we will now carry out an analysis on the realization technique of an ICC profile, using the Gretag Eye-One spectrophotometer. The profile-creation software is ProfileMaker, also created by the Gretag, and the monitor Sony Multisacn E400.

■ **Figure 4.103**
Tools used for the calibration of a monitor.

First of all, all the applications which may modify the reaction of the video card must be disabled, such as Adobe Gamma. Afterwards, it is necessary to keep the monitor switched on for half an hour, to stabilize it. Then the program ProfileMaker is to be started. In the first window one has to specify all the settings, the status one wants to achieve at the end of the calibration.

■ **Figure 4.104**
Starting window of the calibration process.

First of all one has to set the device one is calibrating, in this case a CRT. After the software asks the gamma values, white point and luminosity one wishes to be acquire. One can choose among some predefined values, but there is also the possibility of customizing them.

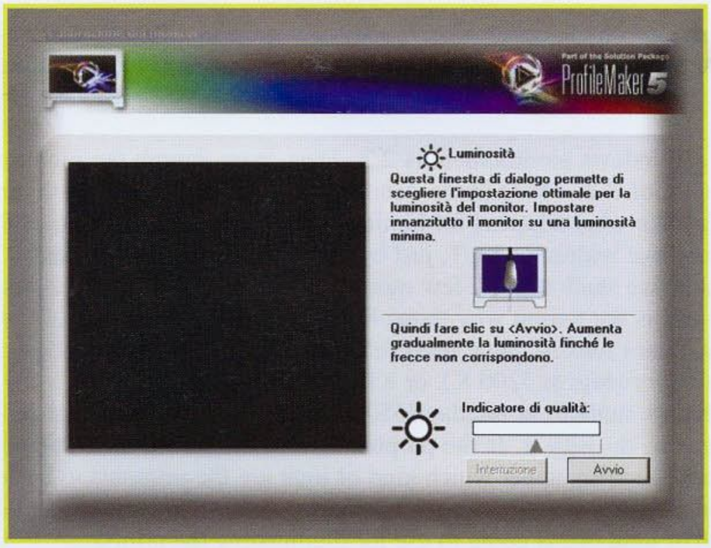
The gamma for a monitor in a Windows environment is 2.2. Thus, even though the monitor has a gamma of 2.5, the calibration brings it to the value of 2.2. The gamma cannot be changed thanks to a hypothetical hardware system directly applied on the monitor, like what happens for luminosity, contrast, and sometimes for the white point as well. The change of the gamma takes place within the LUT, just by operating in the graphic card.
Instead for the white point value, we shall spend a few more words. "White", like all colors, is a feeling generated by the brain, it is not "absolute": this is the reason why terms like "white point temperature", "white balance", etc... exist. For practical reasons some "standard whites" have been defined. Starting from the coldest one to the hottest one: the white generated by a tungsten bulb (white at 3200 K), or a halogen bulb (3400 K), the white corresponding to sunlight in the early afternoon (5000 K), or sunlight at midday (6500 K). How should the white point be set on a monitor? It depends on what the user wants to do with it. If the purpose is obtaining the best correspondence possible with the printouts, another question must be asked: under which kind of light are the printouts to be seen? The balanced light of a photographer's studio, a natural light passing thorough a window with white curtains, and so on; therefore one has to set the closest value to the kind of light one is going to use for illuminating the prints.
But if one does not have control on the light in an environment, which is always the case, it is better to agree on a compromise, choosing a white point between artificial light and natural light. A good choice is white at 5000 K, which has been for many years "THE" standard for graphics firms and is widely used still today. The limit of this calibration is that monitors which are not explicitly built for projecting have a much colder "native white" point; forcing a hotter calibration requires compromises, which limit in turn the dynamic gamma which can be obtained and a slight decrease of the luminance.
The white at 5000 K might be inappropriate if the computer is used not just for adjusting pictures for printing, but for seeing pictures on Internet, or writing texts, and controlling electronic catalogues. It would appear too hot and yellowish. It would be better to choose the 6500 K white point.

Once the target values have been chosen, the real phase of calibration can begin. First of all contrast must be regulated. To do this the calibration program sends the monitor two patches with very similar luminosity, for example RGB=[254,254,254] and RGB = [255, 255, 255]; the instrument reads them and detects whether the monitor is actually able to distinguish them. If not contrast has to be increased with the hardware command on the monitor. The most of the time it has to be set to 100%.

■ **Figure 4.105**
Monitor calibration and the process of sending the patches of maximum luminous intensity for contrast calibration.

Now it is time to adjust luminosity. One starts by setting the minimum value of luminosity with the hardware command on the monitor, then this value is to be increased until the instrument recognizes that the black luminosity increases becoming "dark grey". At that moment one has to turn it back down, slightly decreasing the value to make the black turn back to the darkest value possible.

■ **Figure 4.106**
Display of luminosity calibration process.



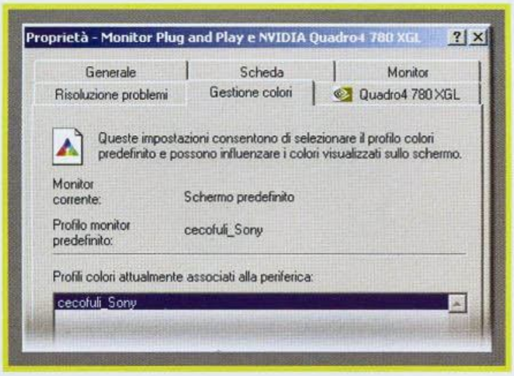
Afterwards one continues with the calibration of monitor white so as to reach the value set at the beginning, for example 6500 K. Some monitors have a hardware command which works directly on the electric cannons. Moving the cursors it is easy to reach the desired values, both for the color-temperature and the white luminosity. Normally these hardware-regulations are not enough for an accurate calibration, so the software has to operate on the software too to get closer to the target values. If no hardware commands are available, the software works on the video card only.

At this moment the software reads a series of colored patches in order to have a complete description of the monitor and it sets which curves have to be loaded on the LUT of the graphic card, in order to reach the values set at the beginning in the best way possible.

Once the computed curves are inserted into the LUT of the graphic card, they are then memorized in the profile which is being created. This way, every time the computer is started, these curves are loaded into the graphic card. In other words, the status created during calibration has to be stable to make the profile reasonable, therefore the exact condition at the moment of the creation of a profile must be restored each time. The hardware controlling commands on the monitor, such as the luminosity, the contrast, etc... do not have to be touched anymore.

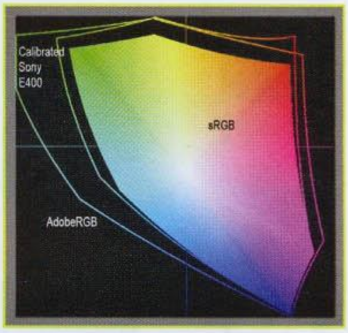
By exiting the software application, one can make sure that the predefined profile of the monitor coincides with the one generated by the calibration and profiling process.

■ **Figure 4.107**
The new profile loaded in the video card properties.



Now the software has built both an LUT correction chart to be loaded in the video card memory for having a first raw correction of the colors, and an ICC profile which corrects the remaining differences, and "describes" the features of the monitor, which may be employed by the ICC-compatible applications.
It is important to keep in mind this double passage: the correction chart for the video card is loaded each time the operating system is started and adjusts the gamma factor and neutralizes the greys, at least in a first approximation.
This correction based on the calibration of the video card, similar to what Adobe Gamma does, is immediately visible and does not require the involvement of a Color Management engine; thus it is usable by all the applications, even the not ICC-compatible, but it is not completely accurate. The ICC profile is able to achieve this task, showing to the Color Management engine, and therefore to the ICC-compatible applications, the residual gaps to be adjusted and the gamut limits of the monitor.

Now the color gamut described by the profile which has just been created has been outlined. The innermost graphic, completely colored, represents the sRGB color-space. This will be soon explained. The outermost graphic represents the AdobeRGB space, while the graphic in the middle describes the gamma of colors available in the monitor which has just been calibrated.



■ **Figure 4.108**
Gamut of the monitor which has just been calibrated.

As can be seen, the monitor is able to cover a wider gamut compared to the sRGB, which is an appreciable result. Without this color profile one would have had to settle for the one provided by the producer.

■ **Figure 4.109**
Comparison between the calibrated monitor and the non-calibrated one.

As can be noticed in this chart, the profile is of less use in the yellow-orange-red area, which is very important for the printouts because it is exactly the zone where the inkjet printing machines have a wider gamut.
The employment of the spectrophotometer Gretag Eye-One has thus returned a correctly calibrated monitor. It has also allowed it to be exploited better by creating a gamut slightly wider in an important range of colors.

Now we shall check the efficiency of the profile just created. The Macbeth 24-notch target has to be purchased for this task.

■ **Figure 4.110**
Macbeth Color checker.

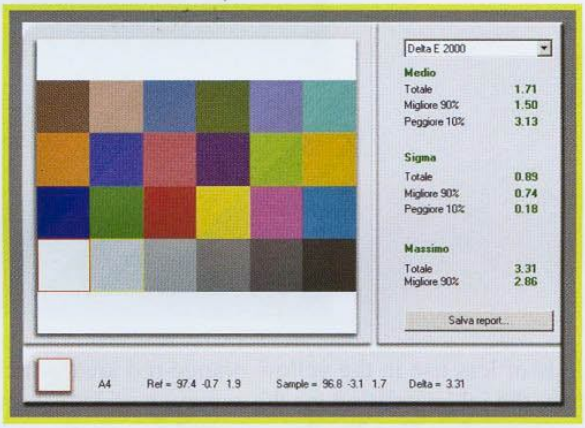
After the patch must be compared with the spectrophotometer, revealing the Lab coordinates of the 24 colors.

■ **Figure 4.111**
Lab values of the Macbeth Color checker.

With Color Picker, which is a module of ProfileMaker, we start from the monitor profile we have just created. For example, the Lab coordinates [38, 14, 16] of the first sample are associated to an RGB value [94, 28, 13]. This operation must be repeated for each notch and thus gaining all the RGB coordinates.

■ **Figure 4.112**
RGB values of the Macbeth Color checker relatively to the profile just created.appena creato.

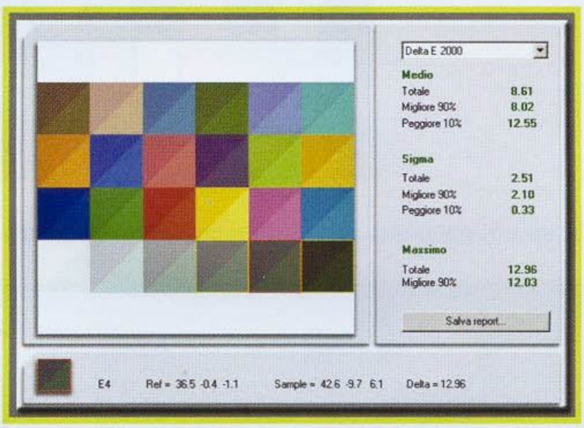
Once the RGB coordinates are sent to the monitor it is possible to measure them with the spectrophotometer. Now we have both the original target Lab coordinates and the ones visualized on the monitor: it is possible to compare them.



■ **Figure 4.113**
The two compared profiles.

Each patch (small square) is divided in two diagonally: the left part is the color which would have been reproduced, the right one the color actually visualized on the monitor. The white patch with red edges is the one with higher Delta E, the least correct. The average Delta E is 1.71, which is, in any case, very low. It is reputed this value does not have to exceed 3 for being considered a perfect calibration. The instrument and the software have done a good job.

Now we will try to use the profile given by the manufacturer (canned profile) together with the monitor.



■ **Figure 4.114**
the profile provided by the manufacturer leads the monitor to an unsatisfactory status.

The mean Delta E in this case is 12.96! The manufacturer has given a profile, but without saying what the status was for the monitor when the profile was calibrated. It could have at least explained how to set luminous and contrast commands, but did not. We also tried to leave the same values obtained before with the instrument, but they did not seem to be right, as all the colors appear too bright. Furthermore, this profile does not correct green dominance, as can be noticed in the green patches in the last row.

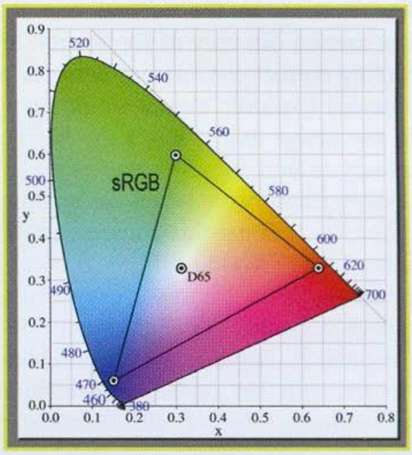
Another system for creating a profile is by using a calibration software, such as Adobe Gamma. In this case the profile is created "on sight" which means that one's eye is the main tool, with all its advantages and disadvantages. It is worth consulting a guide of the software for calibrating monitors with Adobe Gamma. It is not possible to achieve the same precision of a colorimeter. The calibration of a monitor can be improved with programs such as Adobe Gamma, but these methods are not suggested for professional purposes.

WORKSPACE

So far we have discussed RGB coordinates, CMYK, Lab and monitor profiles. But which is the best way of using these tools in practice? First of all notice that a digital picture comes from RGB devices, that is from digicams or scanners. Therefore RGB is the native space of digital images. It has been seen that ICC profiles always refer to a specific device, representing a complete colorimetric description. Furthermore, a conversion from the RGB monitor profile to a CMYK printing machine profile "creates" some CMYK values representing specific colors for that specific printer. In other words, if the same image, converted this way without any changes, is sent to another printing machine with another ICC profile, the printout is going to have different colors, not corresponding to what was seen on the monitor.

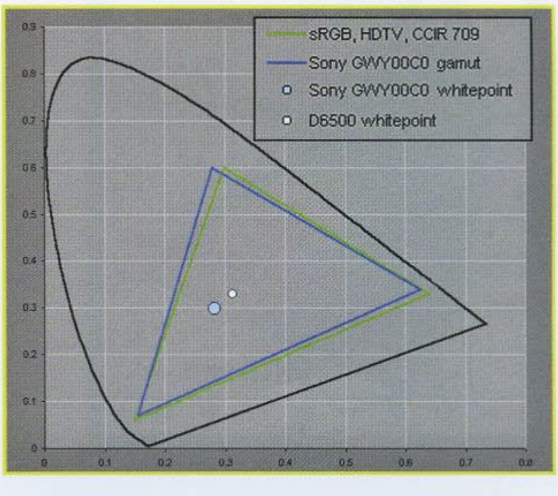
Specific profiles exist which are not devoted to describing the characteristics of some devices, but are used as workspaces. Spaces like sRGB AdobeRGB etc... were born this way. To understand this idea better, imagine having to create a gamut of a hypothetical monitor with the following characteristics:

. Gamma: 2.2
. White Point: 6500°K
. RED chromatic coordinates: ....[0.6400, 0.3300]
. GREEN chromatic coordinates: .. [0.3000, 0.6000]
. BLUE chromatic coordinates: ..... [0.1500, 0.0600]
. WHITE chromatic coordinates: ...[0.3127, 0.3290]



■ **Figure 4.115**
sRGB workspace.

The **sRGB** (Standard Default Color Space for the Internet) was born in 1996 with an agreement between Microsoft and HP, with the aim of producing a standard color space for their systems, both software and hardware, open to all those interested in it. The purpose was using a color definition independent from the device, simple and well-built. In fact, if one pays attention one notices sRGB characteristics are very similar to a traditional monitor. This is not by chance.

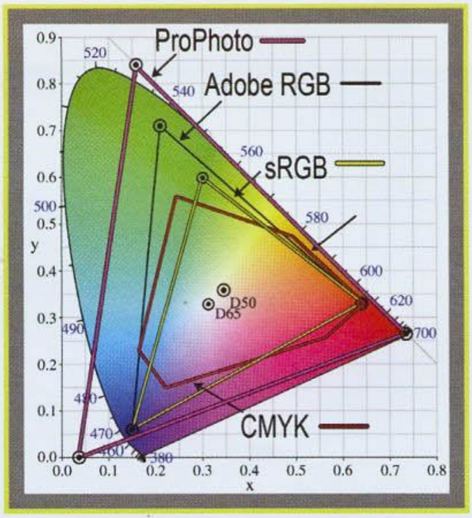


■ **Figure 4.116**
sRGB workspace.

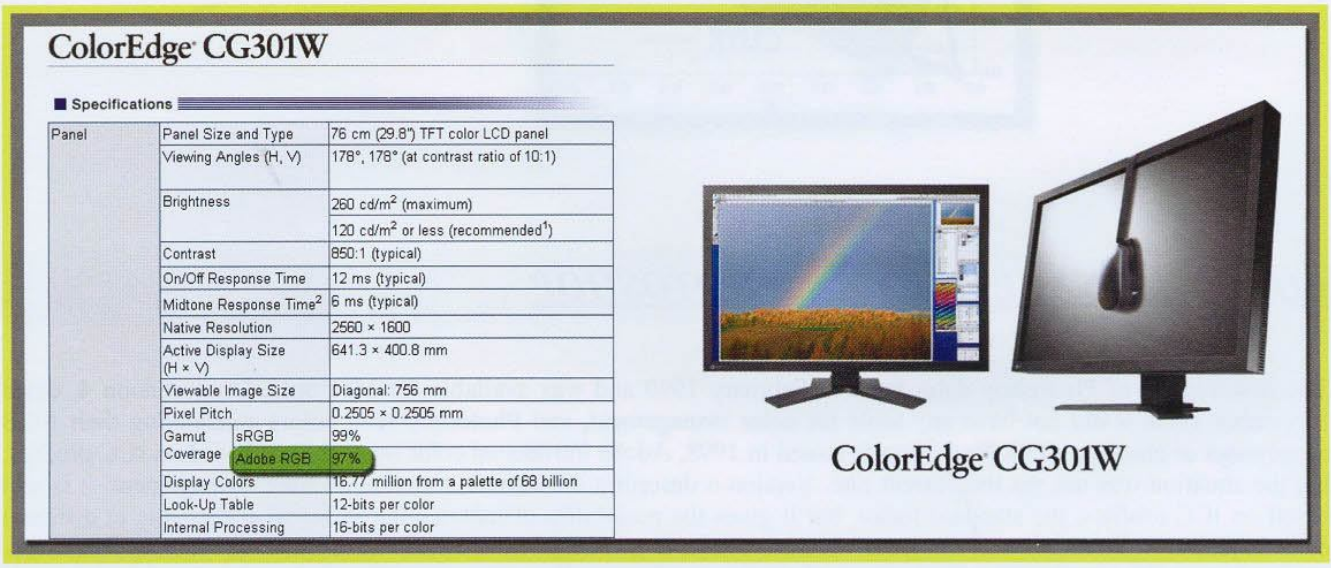
This means that in ICC sRGB profiles, where the famous chart containing all the conversions from RGB to Lab is included, for each RGB value the corresponding Lab one is known. Whenever one talks about sRGB, what is meant is a profile independent from the tool, as it is a profile established at that moment.
As a result, if a real monitor has been calibrated with values Gamma 2.2 and white point 6500 K, the only difference regarding sRGB is the values of phosphors. It is evident that colors are seen reasonably well in an image saved with an sRGB profile, seen in Windows, even with a program not able to manage profiles (like IE or the Windows' viewer). We said "reasonably" because phosphors are different and therefore something allowing one to convert images using two different profiles, like Photoshop, is necessary.

Another feature is present in sRGB. It has a symmetric grey axis. This means that RGB coordinates composed by terns of equal numbers identify the Lab neutral grey color, without any hue. For example if RGB = [128, 128, 128] is the medium grey, then RGB = [23, 23, 23] is a dark grey and so on. This property may seem obvious to those that have never heard about color management and have always worked with RGB coordinates as if they were absolute. In truth it has been noticed that the color identified by the term RGB = [128, 128, 128] depends on the associated ICC profile, that is the Lab coordinates. If your monitor has not been well calibrated before creating the profile, that term might identify a greenish color, whereas the real grey could be associated to RGB = [128, 121, 128]. If one uses the monitor profile as a workspace there would be no certainty of a symmetric grey axis and all the correction tools of the color provided with automatic algorithms, like for example Auto Contrast and Auto Color, would not work correctly. Instead, if one uses the sRGB as a profile of the image to be computed, the colors associated to RGB coordinates are more predictable and are correctly visualized. For example, an image with sRGB profile containing colors with coordinates RGB =[128,128,128] would be converted into RGB = [128, 121, 128] when observed on monitor coordinates, that is the RGB coordinate of your profile corresponding to grey in Lab coordinates, without dominant green, and then it would be sent to the monitor. This change is called "compensation to monitor".
One of the differences between MacOSX and WindowsXP is that in the first case the compensation to monitor is managed by the operating system, in the second one it is not. This means that in Windows environment the compensation to monitor is carried out only at application level. Thus if one looks at a picture with Photoshop, which manages the monitor executing correct conversions between profiles, the picture is seen with correct colors. If one looks at it, for example, from Internet Explorer, colors would be wrong, because, like many other programs for Windows, it ignores the profile within the image and it sends the RGB coordinates to the monitor as they are. To see the colors correctly it would be necessary to find oneself in the lucky case whereby the image has exactly the same profile created for our monitor, which is quite improbable when navigating the Internet. This does not happen in Mac environments: the operating system has the task of doing this compensation for applications like Internet Explorer as well, and images are always viewed correctly. TAKE NOTE: all this is valid if the monitor has been calibrated and an accurate profile describing it has been created. Since the most users navigating on the Web use Internet Explorer in a Windows environment, the sRGB space becomes very important: sRGB is the profile of a medium monitor calibrated with a gamma of 2.2 and white at 6500 K. Anyone who has calibrated their monitor with these parameters will "almost" see correctly all the images created with this profile, even without the automatic conversion. Thus if one has to post images on the Web, it is necessary to convert them to sRGB before. But this space has a fault. It has the gamut of a monitor. A digital or chemical picture has a wider gamma compared to the sRGB. The price to pay for preserving the compatibility with whoever does not use color management for monitors is high: one looses a big range of colors in the original conversion between a digicam profile (or scanner) and the sRGB. Another workspace having a wider gamma has been defined for this necessity. Many other workspaces exist besides the sRGB. One of the most famous is surely the Adobe RGB.

■ **Figure 4.117**
The three main workspaces.



AdobeRGB is a color space developed by Adobe in 1998. It was fulfilled with the aim of covering all the colors obtainable by a four-color process printing machine, using RGB colors. AdobeRGB color space covers almost 50% of visible colors specified by the Lab system. Instead, the sRGB covers 35%.
How are these workspaces used? Everything begins with images, scanned, digitalized, or taken with a digicam. From here onwards, the long process of color management begins. If for example a cheap digicam is used, 99% of the times, without the user knowing it, images are saved with an sRGB profile. This means these images have a gamma of 1/2.2, the inverse compared to the monitor; a white point of 6500 K and RGB coordinates referred to sRGB space. Thus these images can be correctly seen on a monitor, also without the aid of an ICC compatible software and all colors are represented by the CRT. Some photo cameras, more expensive and professional, allow one to save in AdobeRGB and RAW. This means these photos have a wider chromatic gamma compared to the same sRGB. But unfortunately not all of these colors can be seen "on monitor". It is the same thing happening to HDRI images, which contain many colors and not all of them are visible on the monitor. In actual fact, if we choose to believe what some manufacturers say, some LCDs exist which seem to allow a workspace in AdobeRGB, such as those of Eizo's "Color Graphics" series.


ColorEdge CG301W

■ Specifications

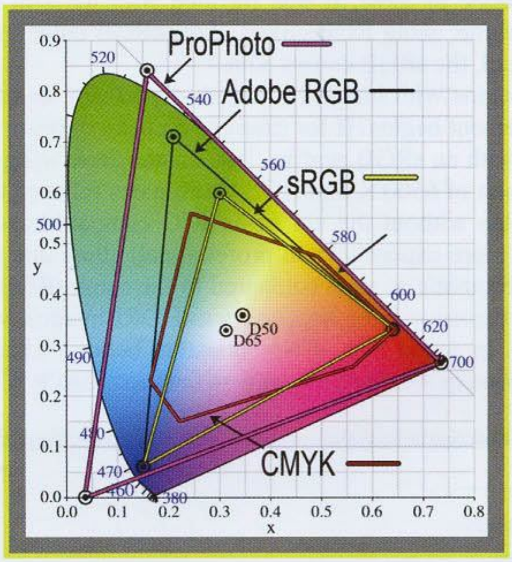
Panel	Panel Size and Type		76 cm (29.8") TFT color LCD panel
	Viewing Angles (H, V)		178°, 178° (at contrast ratio of 10:1)
	Brightness		260 cd/m² (maximum)
			120 cd/m² or less (recommended[1])
	Contrast		850:1 (typical)
	On/Off Response Time		12 ms (typical)
	Midtone Response Time[2]		6 ms (typical)
	Native Resolution		2560 × 1600
	Active Display Size (H × V)		641.3 × 400.8 mm
	Viewable Image Size		Diagonal: 756 mm
	Pixel Pitch		0.2505 × 0.2505 mm
	Gamut Coverage	sRGB	99%
		Adobe RGB	97%
	Display Colors		16.77 million from a palette of 68 billion
	Look-Up Table		12-bits per color
	Internal Processing		16-bits per color

■ **Figure 4.118**
The excellent monitors of Eizo, which allow a chromatic space in AdobeRGB.

But they are very expensive and therefore to be used in the professional field only. As a consequence, some colors are not visible on "traditional" monitors, although they are present in the image, whereas maybe printing machine are able to reproduce them. The task of Color management is to allow a correct correspondence between colors of each device and this can happen only with calibrated devices, ICC profiles and software able to use them. We will cover the part concerning printing, because it would require information going beyond the purposes of this book.
Thanks to the fact that images in AdobeRGB include all that range of colors, even though they are not visible on monitors, it is possible to operate with higher quality on post-production effects, exactly like when using HDRI. By having a wider dynamic gamut one has more chromatisms available for possible corrections or changes. This makes them excellent for printing most of all, since its gamut usually includes that of a printing machine like the CMYK. Instead, for images to be posted on the Web, it is obviously better to save in sRGB. This guarantees a correct vision of colors for whoever uses software with Internet Explorer or non-ICC compatible software.

Many other spaces exist in addition to this one, such as **ProPhoto** for example, born from Kodak. With this Kodak decided to abandon the usual concept of monitor and created a workspace tailor-made for pictures. It has been noticed it is useless to use a profile connected to properties of a monitor for digital images. It is more convenient to create a profile which better suits digital cameras' or scanners' needs, for example by defining a very wide gamma which can easily contain colors reproducible by electronic sensors.

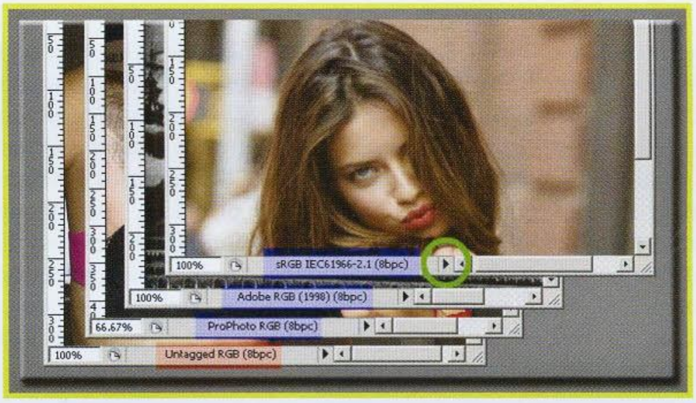
■ **Figure 4.119**
ProPhoto is the widest work space.



COLOR MANAGEMENT IN PHOTOSHOP

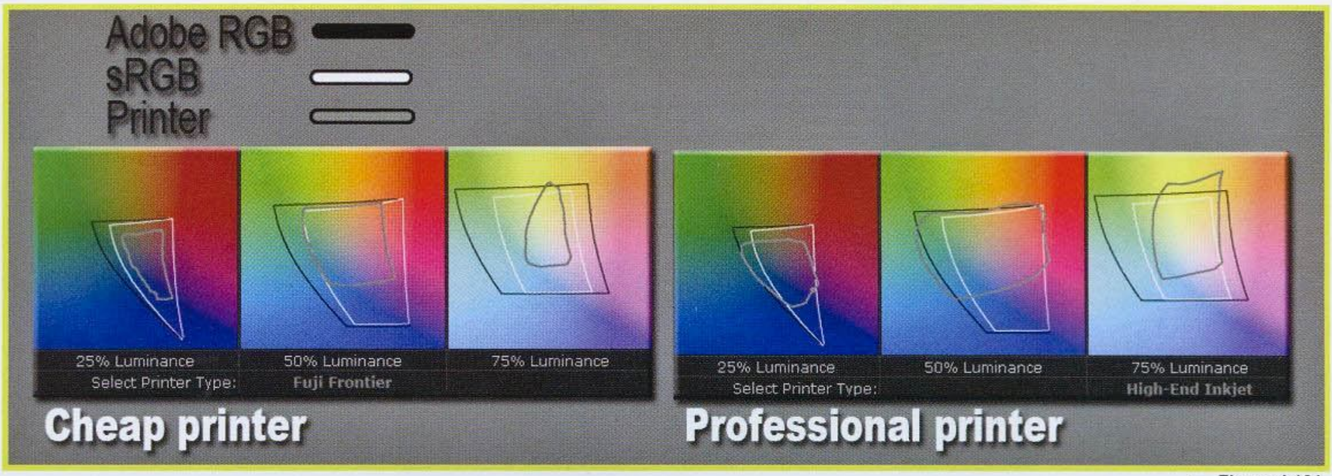
The first version of Photoshop dates back to February 1990 and was available for Mac only. Until version 4, dated November 1996, it did not have any tools for color management, and Photoshop read colors considering their RGB coordinates as absolute. With Photoshop 5, issued in 1998, Adobe introduced color management based on ICC profiles, but the situation was not yet the current one. Version 6 describes the current situation of color management: it is still based on ICC profiles, the standard today, but it gives the possibility of maintaining different documents in different color spaces.
The profile can be saved in an image as well. Images might also lack this possibility, and in this case the image is Untagged RGB.

■ **Figure 4.120**
To visualize the kind of profile associated to the image one just has to click on the black triangle highlighted in the image. A menu appears where it is possible to choose the option Document Profile.



Furthermore, if a new image is created, it is possible to specify the color space to be used. These new observations lead us to define the color spaces sRGB and AdobeRGB. Which one should be used ? The first image might be AdobeRGB, because it is provided with a wider gamut and allows one to compute images with wider chromatic gamma. Generally this is true. Some cameras allow one to choose between sRGB and AdobeRGB. It is advisable to always choose AdobeRGB. But the main question to be asked is: which is the best use of the image? Should it be printed or just displayed? If the picture has to be printed it is advisable to keep AdobeRGB, whereas if it is displayed, it is better to change profile and pass over to sRGB.
A printer has some colors that the sRGB does not have and usually AdobeRGB contains almost always the gamut of a printing machine in a four-color process. If the same image is printed before using sRGB and then AdobeRGB, the sRGB image has less shades in blue/green colors compared to the AdobeRGB one. This happens because the sRGB profile has these colors "cut" off, as its gamut is smaller than that of the AdobeRGB.

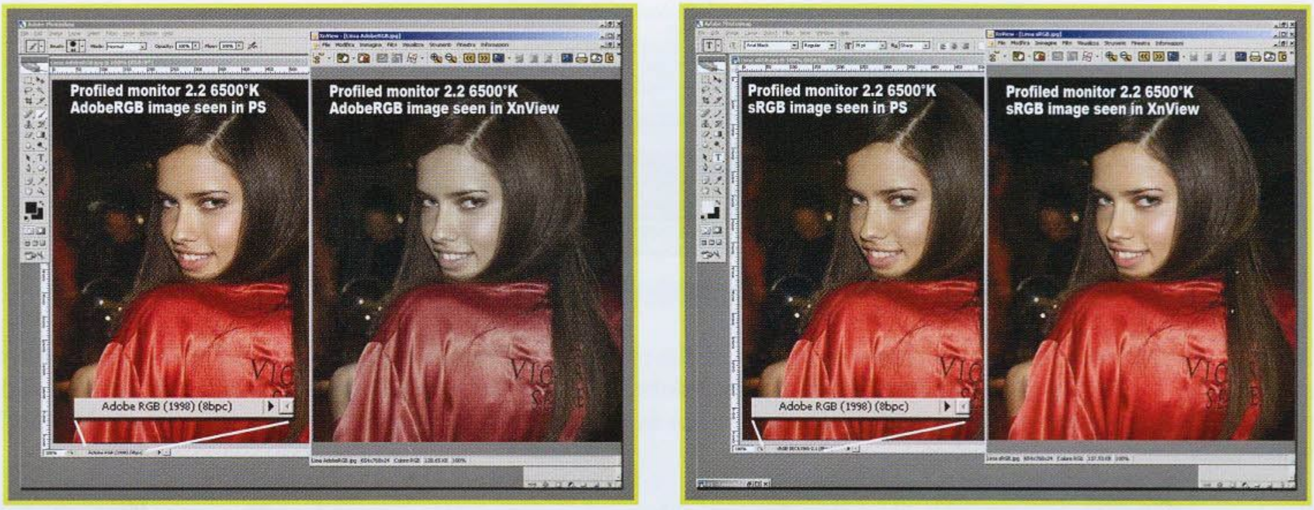
Below the gamut of a cheap printer and a professional one relatively to the same image converted in AdobeRGB and in sRGB.



■ **Figure 4.121**
Comparison between the sRGB and the AdobeRGB gamut, and two CRT printers.

In the case of a cheap printer, which has a noticeably lower gamut compared to a professional one, the difference between the sRGB and the AdobeRGB is little (the gamut of the printer is always contained in both of them). Instead, in the case of a professional printer, it is possible to notice how, with luminance at 50%, a wide range of colors would be cut off, and it would be possible to save them only with AdobeRGB.

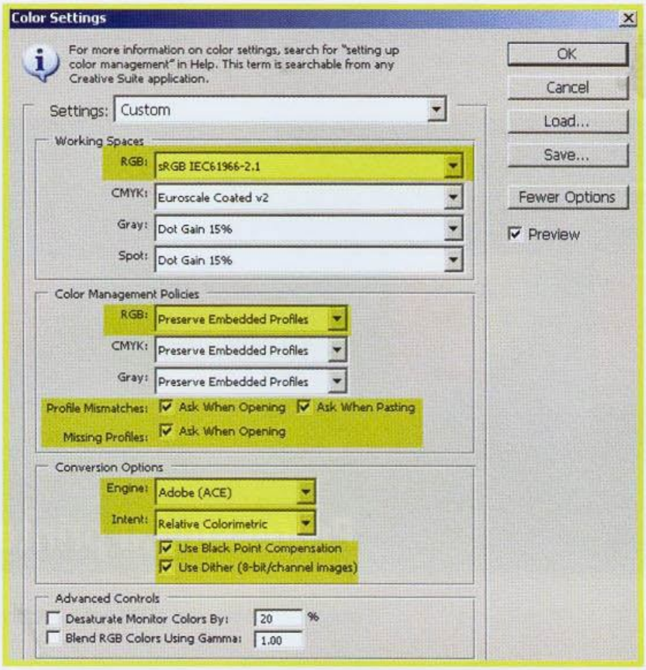
If the images are only displayed, the employment of the AdobeRGB space is useless: monitors allow only an sRGB representation, except for rare cases of professional monitors. Furthermore, an AdobeRGB image would be correctly displayed (supposing the monitors are ALWAYS well calibrated and profiled) only within compatible ICC programs. When these images are visualized outside of these programs, they are always considered as sRGB images and cannot be correctly observed.



■ **Figure 4.122**
The sRGB image, both in Photoshop and in a common visualizer, is correctly displayed. This happens when using the AdobeRGB profile.

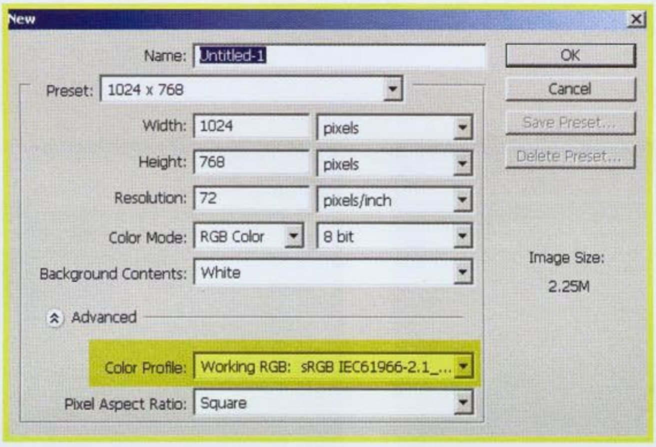
The management, the heart of Color Management, is located in a very complex tool in Photoshop called Color Setting. One finds it through the path EDIT -> COLOR SETTING, which can be activated thanks to the shortcut CTRL+SHIFT+K. Every parameter belonging to it is to be generically outlined, focusing the attention on the parameters highlighted in yellow, those concerning the viewing of images on the monitor.

■ **Figure 4.123**
Photoshop's Color Setting.



With the Setting parameters it is possible to load, with a pull-down menu, some default presets, save some new ones and recall already saved presets.
The next section, Working Spaces, is the one allowing one to choose a work space for every possible color method: RGB, CMYK, grey scale and all the Flat Hues. The predefined profile is, for example, the one that is always opened when images without an incorporated profile are opened. In the case of a brand new image, the definition of the profile is carried out by means of the chart appearing at the moment of the creation of the new file via the option "New".

■ **Figure 4.124**
Inside the display for the creation of a new file it is also possible to assign the profile.



At this point one should have a calibrated monitor, with the profile describing its features. But why is Photoshop able to show images correctly with incorporated profiles, like AdobeRGB or sRGB, while a normal visualizer is not able to do so? What happens when a file is opened? The secret lies in a special procedure, named "Compensation to Monitor", which takes place within Photoshop.

When an sRGB image is opened, Photoshop immediately carries out a color conversion. This operation, which is invisible for the user and does not modify the image, cannot be deactivated, and allows a correct vision of the image in its profiles. The monitor compensation changes the RGB numbers of the image so that the colors seen on the monitor (that is in the space defined by the ICC profile of the monitor) coincide with those the author of the image wants (that is in the space defined by the incorporated profile).
In practice, Photoshop carries out monitor compensation by writing in the video card not the values of each single RGB pixel, but the modified ones. The absolute values of single pixels are preserved. To better understand this concept let us do this simple example.

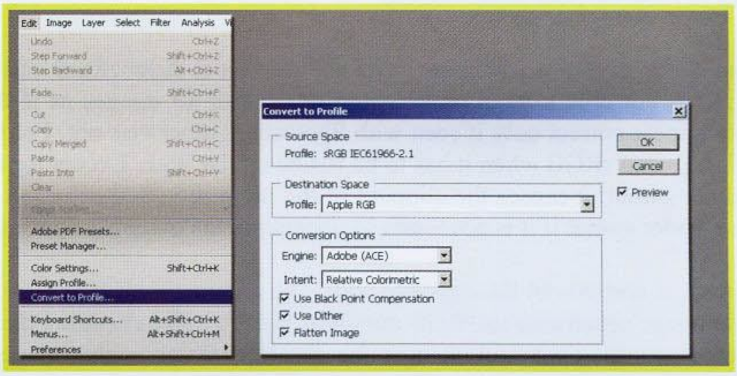
The sRGB image is a simple rectangle, colored with RGB = [128, 128, 128], which corresponds to the Lab coordinates [54, 0, 0]. Photoshop knows this and also knows that this Lab coordinate is displayed on the monitor with the RGB=[128, 121, 128], thanks to the profile created during calibration.

The first thing one might think of doing is to change the value of the pixels from RGB = [128,128,128] to RGB=[128,121,128]. This would change the image physically in an irreversible way. Moreover, the more people observed this image, each time a change would affect the image. In truth Photoshop carries out a monitor compensation writing in the video card of the computer the modified values RGB = [128,121,128], and not the values RGB=[128,128,128]. This way the "B" user can see on monitor the same color the user "A" was seeing on his monitor. And this happens without modifying the image at all. The pixel always remains RGB = [128, 128, 128]. A nice comfort!

Before going on, a distinction between "converting" and "assigning" a profile must be carried out, by understanding how these actions are managed within Photoshop. These commands are named "ASSIGN PROFILE" and "CONVERT PROFILE", and they are contained in Photoshop's MODIFY menu.

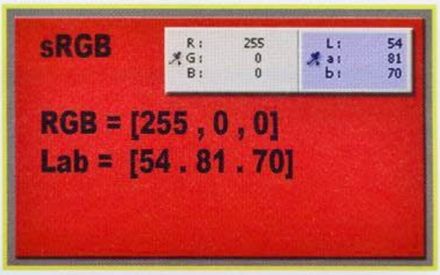
CONVERT A PROFILE

Using the command MODIFY -> CONVERT IN PROFILE another profile can be assigned to the image, changing the number of RGB pixels of the image in order to obtain, for what is possible, the same Lab colors of the non-converted image. With this operation the numbers change in such a way to keep the perceived colors the same. Variations depend on the two gamuts and on the rendering intent chosen: the other changes can be controlled by activating the button "Preview".
The source space is the image space. The destination can be chosen among all the ICC profiles in the appropriate pull-down menu. The color engine has to be chosen as well (Model), and so does the rendering intent.



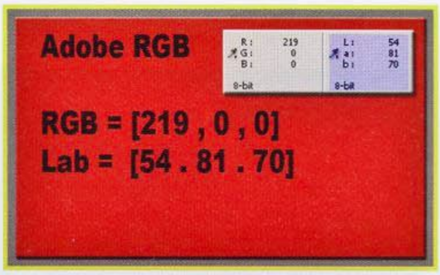
■ **Figure 4.125**
Display of conversion.

Here is an example to clarify this idea. Create a new document with a red colored patch RGB = [255, 0, 0] in an sRGB space. The Lab coordinates, detectable with the dropper in Photoshop, result in Lab = [54, 81, 70].



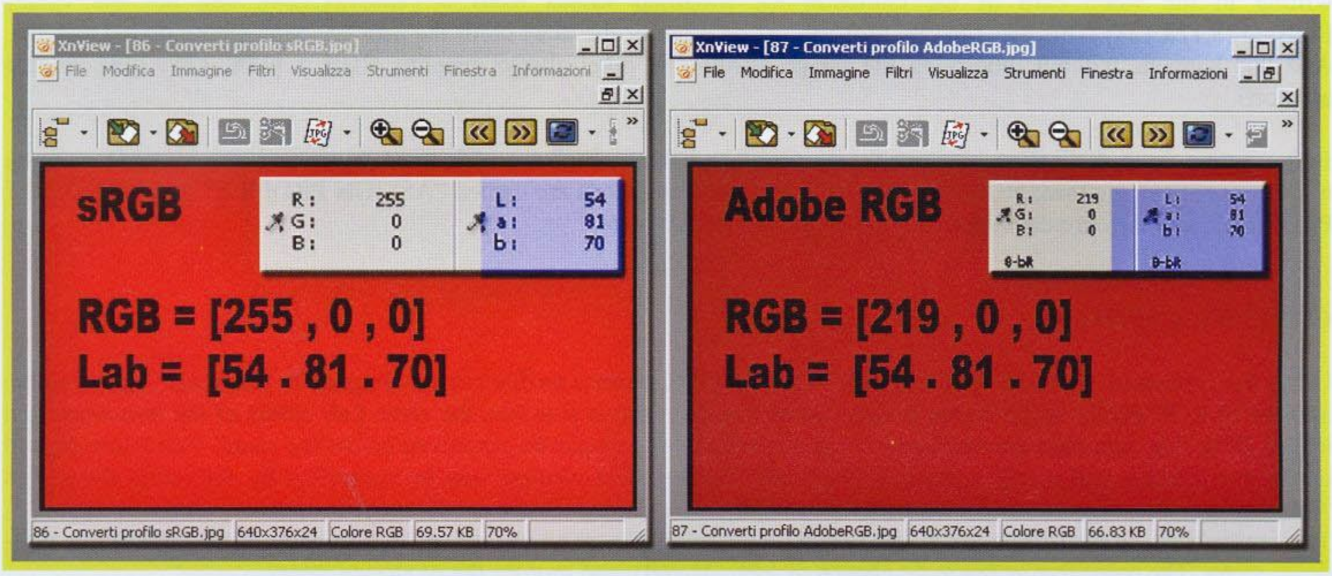
■ **Figure 4.126**
A red patch in sRGB within Photoshop.

Now convert the image profile in AdobeRGB.



■ **Figure 4.127**
A red patch in AdobeRGB displayed in Photoshop. The RGB values are changed, but not the Lab ones.

If pixels are analyzed with the dropper, it will be noticed that RGB coordinates have been modified in RGB=[219, 0, 0], while the Lab coordinates have been kept the same. This happens because monitor compensation is applied in Photoshop. The patches appear differently if seen outside Photoshop.

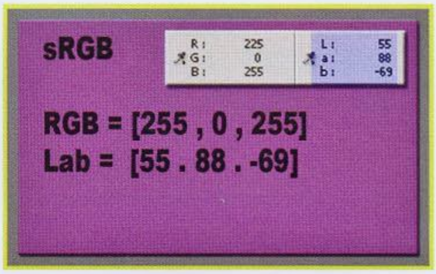


■ **Figure 4.128**
Comparison between the sRGB and the AdobeRGB profile outside Photoshop.

A case whereby profile conversion occurs is when one has a picture shot in AdobeRGB and one wants to issue it on the Web. As we already said, IE is not able to read profiles and deals with all images as if they were sRGB. An image issued in AdobeRGB would appear correct only if seen with Photoshop, otherwise not. For this reason it is advisable to convert the profile of any image into sRGB when it has to be issued on the Web. Except for this case it is unadvisable to convert to a profile with lower gamut: it causes the clipping of the more saturated colors. It might make sense, instead, to convert to a profile with a wider gamut if it is necessary to operate on the picture's contrast and saturation.

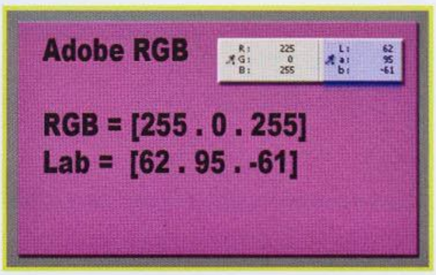
The assigning process, instead, is carried out the opposite way. The command MODIFY -> ASSIGN PROFILE allows one to assign a profile to the image preserving its sRGB numbers, but therefore changing the Lab color. It is a command to be used in specific cases, since this operation changes the absolute values of the image, that is Lab colors. In this sense, it is a more "destructive" operation compared to the previous one. An example of a case where this operation might be useful is when one takes an image without a profile from Internet. In 99% of the cases it is an image with an sRGB profile, therefore it is important to assign the correct profile. In this case we shall make an example using a magenta patch.
First of all create a new file in sRGB, coloring it in RGB = [255, 0, 255]. These coordinates correspond to Lab coordinates [58, 88, -69].

■ **Figure 4.129**
Magenta patch in sRGB inside Photoshop.

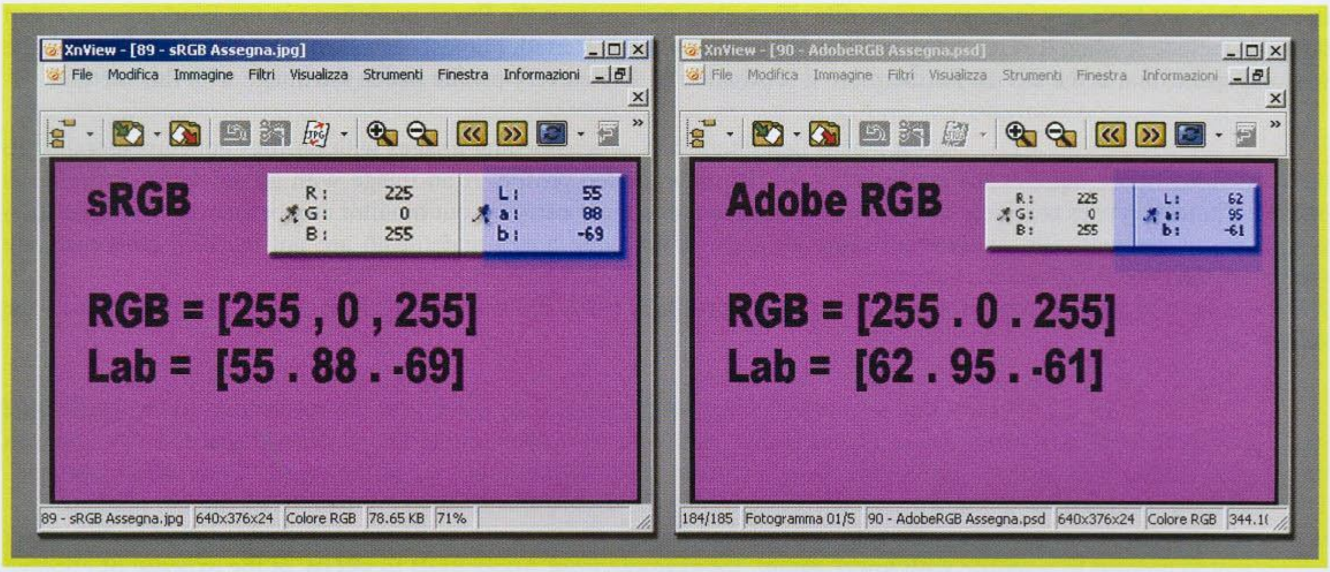


Afterwards assign an AdobeRGB profile.

■ **Figure 4.130**
Magenta patch after the assignment of the Adobe profile inside Photoshop.

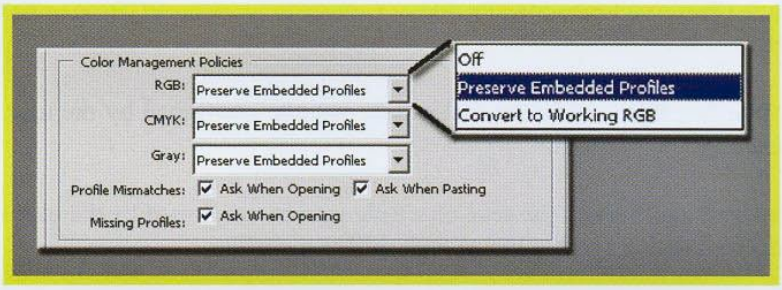


It is evident that the RGB values are the same, whilst the Lab coordinates have been changed. Photoshop changes the image so as to make it correspond to the monitor's profile. This always happens for monitor compensation. But in this case, if one observes the images outside Photoshop, they appear the same. This happens because Windows XP "thinks" in absolute terms and does not consider ICC profiles.



■ **Figure 4.131**
Comparison between the profiles sRGB and AdobeRGB outside Photoshop.

What happens when one opens an image which does not have profile, or when an image has a profile different to that of default, or, moreover, when one pastes from an image to another and the profiles of the two images are different? Thanks to the section Color Management Policies it is possible to decide how Photoshop should behave, or, better, it is possible to allow Photoshop to make all the right decisions or allow the user to decide.



■ **Figure 4.132**
The important Color Management section in Photoshop.

The default settings are those represented in the image below. These settings do not carry out any change in the image and do not require the user's intervention. Many people never realize what happens every time an image is opened with Photoshop, until now. This because actually nothing happened! Maintaining the incorporated profile guarantees the image remains intact and keeps any color management intervention activated. Furthermore, the fact that nothing is ever demanded from the user makes this even more invisible.
For the moment only RGB management is to be analyzed. The pull-down menu concerning this value allows one to choose among three options:

. Off;
. Preserve Embedded Profiles;
. Convert to working RGB.

These three options tell Photoshop what to do when a new image is opened.

Off - it disables the control of the input-image profile; thus when Photoshop opens an image with an incorporated profile, it uses it for the representation on the monitor or includes it during the saving of the file. If the incorporated profile does not correspond to the workspace, Photoshop advises the user with a dialog window. If the image to be opened does not have any incorporated profile, Photoshop assigns to it the Space predefined in the Working space, but does not include it when it is recorded on disk. When a new job is started, Photoshop makes the space coincide with the working space, that is the sRGB. Furthermore it is possible to delete every profile inside the image with this method.

Preserve Embedded Profiles - it means that Photoshop, when opening images with profiles not corresponding to the predefined Space, applies the conservation of the incorporated profile. If an image with an AdobeRGB profile is opened, Photoshop uses precisely this one as working space, simply carrying out monitor compensation. This operation is carried out automatically, without advising the user, if the three boxes underneath are not marked, otherwise any successive decision is delegated to the user.
If a file is opened without profile, the Working space profile is used. If a new document is created, Photoshop uses the default one.

Convert to working RGB - this means that Photoshop proposes as a solution the conversion to the predefined space, usually the sRGB, when it opens images with a profile not corresponding to the predefined space, for example an image with AdobeRGB profile. This conversion is carried out with the intent set in the panel "Conversion Options". This operation is carried out automatically, without advising the user, if the three boxes underneath are not marked.
If special settings are not necessary, the setting suggested for the photographer is Preserve Embedded Profile. It is never convenient to convert to a profile with lower gamut, for example from AdobeRGB to sRGB, at the expense of clipping the most saturated colors. Instead, it might make sense to convert to a profile with wider gamut, in the case whereby it is necessary to work on the contrast and the saturation of the picture.

On the bottom of the panel three other boxes to be marked are found, for choosing when the set Management Colors Rule (Preserve Embedded Profile or Conver to working RGB) is to be confirmed by the user or it is automatically executed...

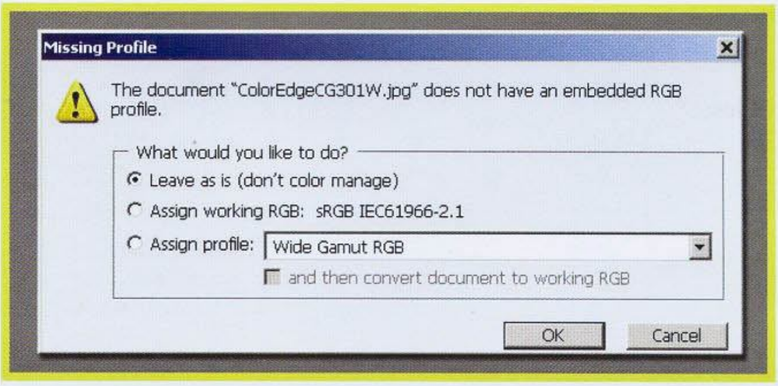
Profile Mismatches - On the bottom of the panel three other boxes to be marked are found, for choosing when the set Management Colors Rule (Preserve Embedded Profile or Conver to working RGB) is to be confirmed by the user or it is automatically executed.

Missing Profiles - by marking this box the set Color Rule is to be confirmed by the user when an image without incorporated profile is opened.

Ask when Pasting - by marking this box the set Color Rule is to be confirmed by the user when a Copy/Paste of an image, or part of it, is carried out within another image with different profile.

The suggested configuration is the one with all three boxes marked. This way Photoshop opens an advice-window whenever the profile of the opened image is absent or not corresponding to the predefined Space. It is possible to confirm the set Color Management Rule with a simple "ok", or choose a different operation among the possible solutions shown.

Now the three advice-windows, and all the alternatives proposed will be illustrated. The first case is opening an image without a profile. In this case, if the option "ask before opening" is active, the following chart appears:



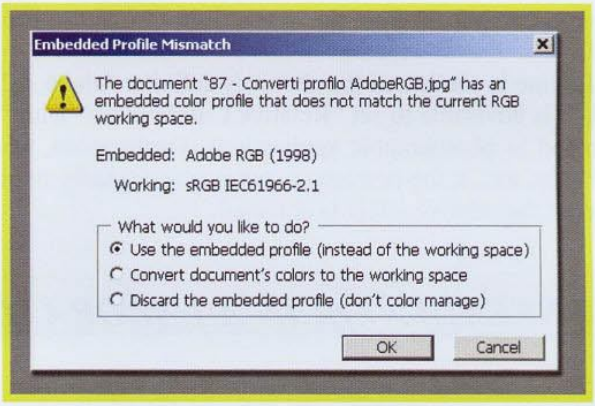
■ Figure 4.133
Photoshop's message when an image is opened without profile.

Do not change - with this option one tells the program to not include any profile, if not specified further on during the saving. It is obvious that Photoshop needs to use a reference color space for representing the image on the monitor, and it uses the default one. In actual fact it preserves the numbers of the colors. If the working space is changed, the RGB numbers are kept the same, but not the color. In other words Lab values are modified according to the new working space. This option is not advisable. It is always better to include a profile in images. This option may be used, for example, when an application external to Photoshop has computed an image (created with Photoshop) saving it without including the profile.

Assign working RGB - it assigns the predefined Space to the image, in this case the sRGB. The program preserves the RGB-color numbers, reinterpreting them in Lab values in order with the work space. If the work space is changed, the profile is preserved. If the assigned profile is not marked in the dialog window it will not be saved.

Assign profile then convert to the work space - it is possible to assign a source profile, for example that of the scanner, or the digital camera, in order to give a sense to the RGB color numbers or to create a starting point for a correct conversion, and convert it directly in the current work space. The work profile is not going to be incorporated during file saving, except when the option is marked in the dialog window. The classic example is an image deriving from linear scanning, to which the scanner profile is assigned. After, if the option "then convert the document to work RGB" is activated, the conversion to the work space is done: in this case sRGB.

The second case is when an image not having a profile corresponding to the predefined one, that is the sRGB, is opened. There are three possible choices:



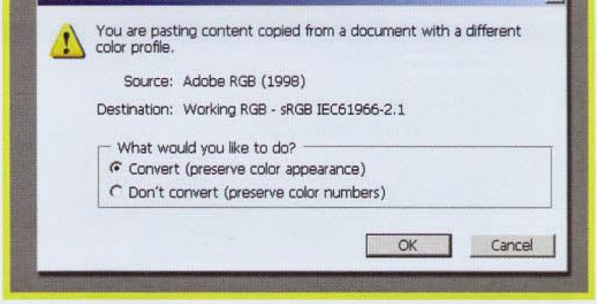
■ **Figure 4.134**
Photoshop message when an image having a different profile relative to the work space one is opened.

Use the incorporated profile - it ignores the missed correspondence and keeps the profile incorporated: in this case AdobeRGB. This is the most probable choice, since it works on the image in the same space in which it has been created.

Convert document's colors to the work space - it carries out the conversion from the incorporated profile to the predefined Space (in the example AdobeRGB and sRGB).

Delete the incorporated profile - it deletes the incorporated profile assigning to the image the predefined space, in this case the sRGB, but it does not incorporate it until it is saved. This option is used for computing an image from a "numerical" point of view. The colors displayed do not make much sense.

The last case is when one tries to copy an image or part of a picture with a different profile.



■ **Figure 4.135**
Photoshop message when one tries to insert an image with a different profile other than to the destination profile.

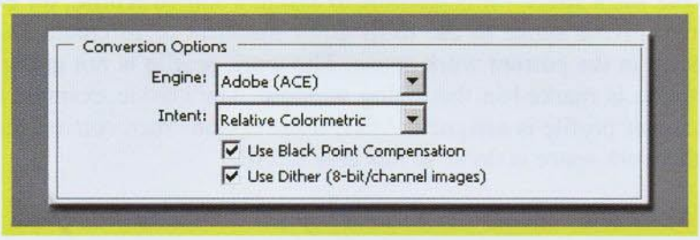
In this case two possible solutions are suggested:

Convert - it carries out a conversion from the source profile (in the example from AdobeRGB), to the destination profile sRGB. This is the most used option, as the RGB values of the image to be pasted are modified in such a way to preserve the correspondence of the colors with the destination image as much as possible.

Do not convert - it does not convert to the destination profile and therefore it keeps the RGB values unchanged, but the colors are going to be distorted.

The last panel of Photoshop's CM tool concerns the rendering intents and the engine used for the computation.

■ **Figure 4.136**
Settings for the intents and the conversion engine.



Conversion options - these two settings, conversion engine and conversion intent, are going to be used by Photoshop in the notification windows "Embedded Profile Mismatch" and "Paste Profile Mismatch". The conversion options do not have an effect on conversions carried out from the menu "Edit/Convert to Profile", where it is possible to choose once by once Engine and intent.
As far as the Conversion Engine is concerned the choice is definite: Adobe ACE is to be preferred in most cases. As for conversion intent, instead, it is advisable to set "Relative Colorimetric" with "Use black point compensation". It is the most used conversion method in photographic applications. Furthermore, when one converts to an RGB work space (sRGB, AdobeRGB, ProPhoto, etc...), the perceptive intent is not usually implemented in the profile, and therefore, by selecting it, the same effect of the relative intent is obtained.

PRINTING ON ONES OWN OR IN A LABORATORY

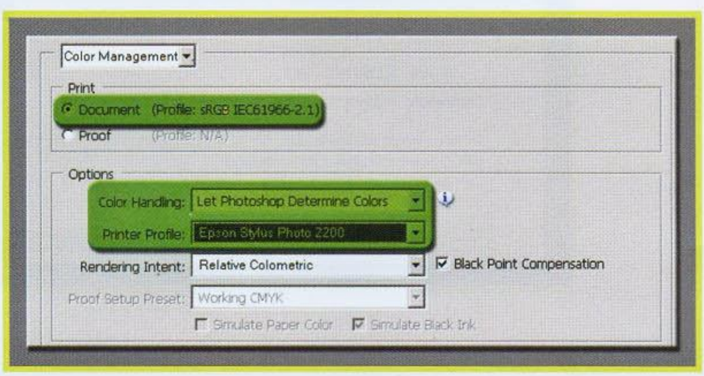
PRINTING ON ONES OWN

Printing by ones own may lead to the best results in terms of gamut, especially with the modern ink-jet printers, and allows an excellent predictability and repeatability of the result, provided that an accurate profilation has been carried out. It is reminded that, for each printer, an ICC profile is needed for each photographic paper used. Furthermore, the possible change of the ink, especially if the brand is changed, requires the creation of new profiles. Sometimes the profiles furnished with the printer are valid enough, but normally they refer to a limited set of papers, and, obviously, to the original inks only.

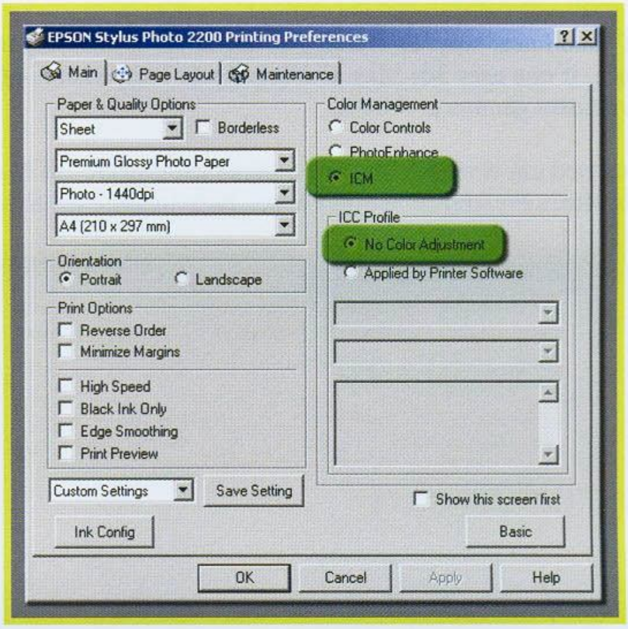
To print correctly essentially two operations must be carried out:

- Conversion from the image Space to the printer/paper/ink destination profile. This conversion, to be carried out with the selected Intent, has to be directly done in Photoshop's printing interface. In the example here below the menu "File/Print with Preview" has been opened in Photoshop, and both the printer profile (an Epson Stylus Photo 2200) and the conversion intent have been selected.

■ **Figure 4.137**
Management of the printer and its profile in Photoshop.



- Once the profile has been set and the other parameters have been checked (size and orientation of the image) one proceeds by confirming with "Print…". Now one enters the interface of the Print Driver provided by the constructor and, as well as setting the kind of paper used, one has to deactivate the "Printer Color Management", in order to avoid the driver from intervening in any way on the image, and making it only convert from RGB to CMYK and send data to the printer. Unfortunately, the position of the command for the deactivation of the "Printer Color Management" is not standard: it must be found among the numerous options in the Driver interface. In the case of the Epson Stylus Photo 2200 the commands for color management can be found in the Advanced section of the printer control-panel. More precisely, the option ICM has to be marked, and then the box No Color Adjustment also, since Photoshop controls the whole CM process.



■ **Figure 4.138**
Epson Stylus Photo 2200 control-panel. The color management is turned off.

LABORATORY PRINTING

Printing on ones own surely is the higher-quality solution, and the one which involves the user the most, as was the case with the dark room of old. But unfortunately the costs for printers, paper and ink make this choice quite expensive. Therefore the user often chooses to commit the print to a Laboratory.
Laboratory prints are carried out on the classic color chemical-paper by means of complex and expensive printing machines. Since one can count on a Laboratory for managing ICC profiles, the obtained printouts are of very high quality. The only fault concerns the gamut, which is more limited compared to an ink-jet print.
The problem arises when the Laboratory does not use ICC profiles and does not guarantee the exclusion of all those automations (exposure, contrast, dominant corrections, etc...) which alter the image in an unpredictable way, making all the management carried out from the beginning useless.
The ideal condition is when the Laboratory carries out color management via ICC profiles, therefore being able to deliver the image with the original color profile (eg. ProPhoto and AdobeRGB). In the most cases, instead, when the Laboratory does not give precise hints, it is worth converting to the most common profile which is the one with which they should be more aligned: sRGB.

CONCLUSION

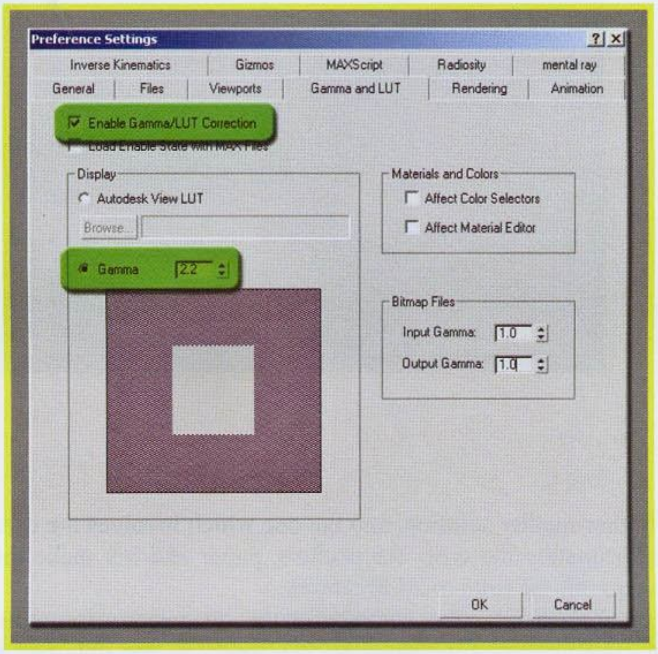
The ICC profile is an excellent tool for managing the color in digital photography correctly and Photoshop is a photo re-touch application which implements this technology very well.
The best condition is obtained when profiles are obtained made especially for the owned devices and when these profiles can be updated periodically to record possible efficiency oscillations by which any hardware is necessarily affected. Certainly this system is costly both in terms of time and money because of the purchase of software/hardware necessary for the operations of calibration and profilation. However, abandoning to color management means literally frittering away one of the biggest potential qualities the digital system offers: the complete control on an image.

LINEAR WORK FLOW (LWF)

This long discussion on the Color Management System has been carried out for a specific reason: to understand the LINEAR WORK FLOW, which is the system used for the linear management of a rendering. This might appear to be no big deal, but it can change the way of fulfilling ones renders if well understood.

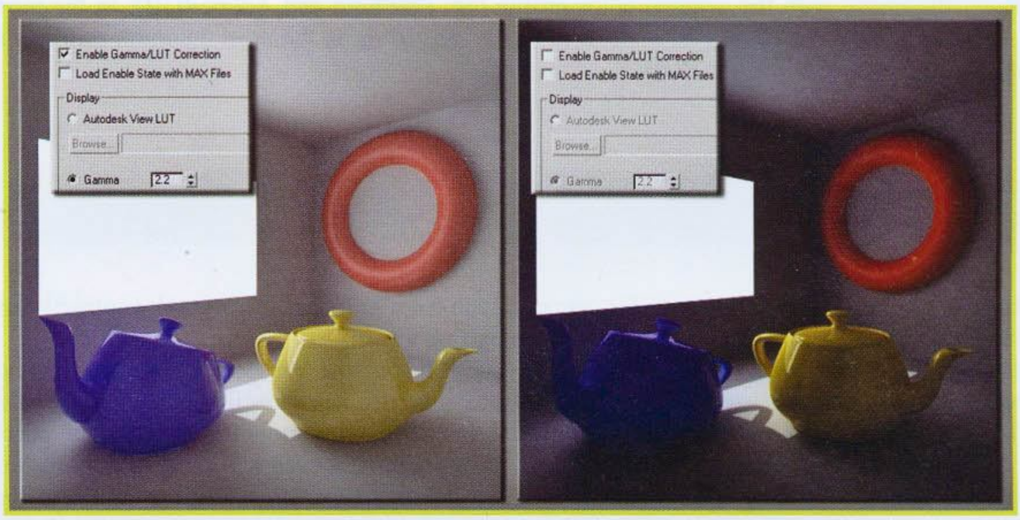
Monitors do not visualize images in a linear way, but they have a gamma factor, which may vary from one monitor to another. The calibration leads the screen to a known condition, that is, in most cases, it sets the value of the gamma factor to 2.2.
Instead, 3ds Max "thinks" in a linear way. Everything concerning it, including textures, colors and renders, are treated as linear. As a consequence, if one uses 3ds Max as a non-linear tool, what is produced is visually not correct. This in turn leads to renders with evident gamma problems.

The first operation to be carried out is modifying the gamma factor inside 3ds Max. to achieve this one has to select the Gamma pannel within 3ds Max's Preference Settings panel, by activating the following options.



■ **Figure 4.139**
Change of gamma within 3ds Max.

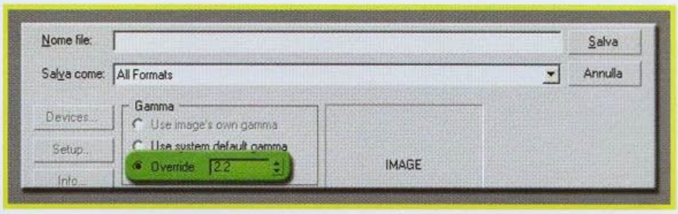
This way images and whatever is to computed will be linearly visualized. In truth the Gamma 2.2 which has just been modified represents the inverse of monitor gamma, 1/2.2. This process happens in the same way in digital cameras: to counterbalance the monitor gamma an image with inverse correction is realized. The effect is linearization of the displayed picture.



■ **Figure 4.140**
Differences in the visualization of a render with or without the activation of 3ds Max's gamma.

In the scene above, ***VRay*** has been set to default values, this means with ***Dark*** and ***Bright multiplier*** set to 1 in the ***Color mapping linear***. Typically (as seen in the right-hand image), for improving the luminosity of the scene, one would have increased the intensity both of the sun and the ***skylight***. Consequently, the areas more exposed to direct light would have been burned, and rendering times would have been increased because of the higher intensity of light. Otherwise one might have changed the ***Color mapping***, by using the ***Exponential*** option. Instead, it has been enough to modify the visualization of the gamma in 3ds Max.

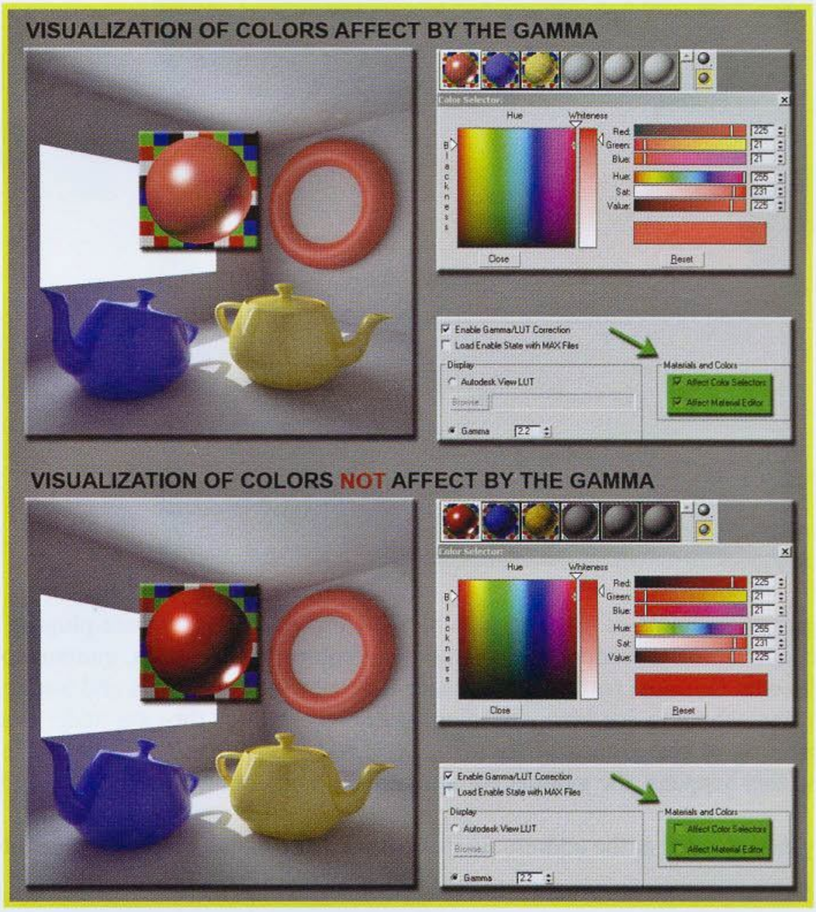
It is necessary to pay attention to one thing: 3ds Max is only showing the image with **Gamma 2.2**, but it is possible to observe this visualization only in 3ds Max or in ICC-compatible programs. When one has to save, this visual information goes lost. To avoid this, the ***Override*** option has to be activated. This way the 3ds Max gamma is "inserted" into the image. This is if one wants to save in ***LDRI*** format: .jpeg or .bmp. If one wants to save in ***HDRI*** format, programs like Photoshop recognize the correction applied to the gamma factor, and visualize the image correctly. If the same image is instead observed outside Photoshop, as with XnView, it would appear dark.



■ **Figure 4.141**
Employment of the gamma-modify option during the saving of a render from the Frame Buffer of 3ds Max.

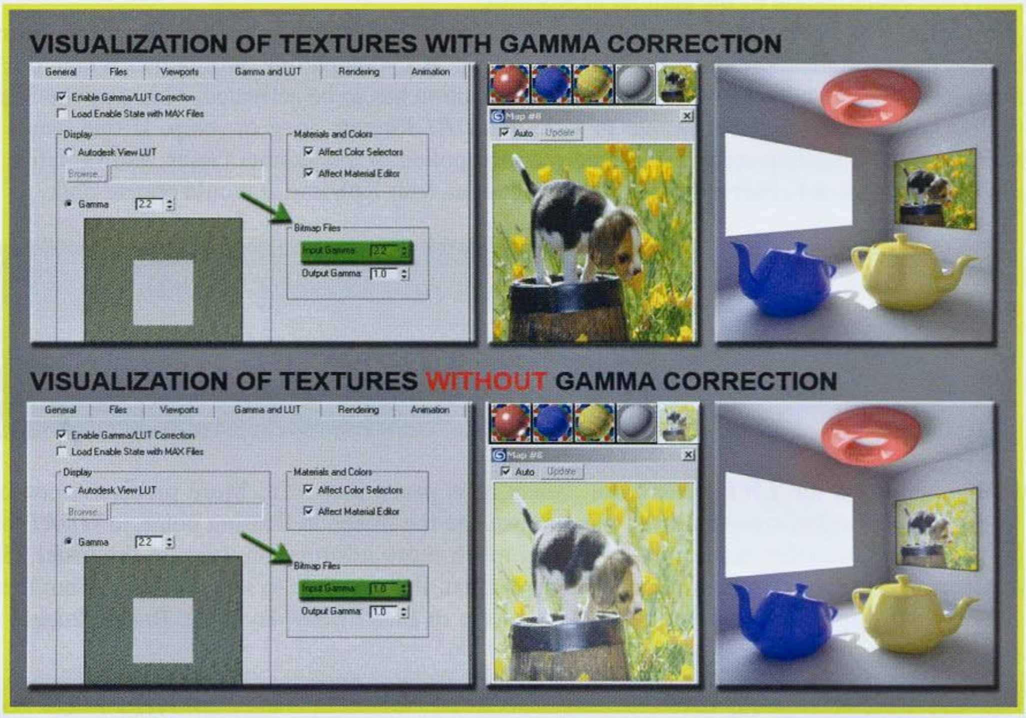
But this is not yet the ***LWF*** system. Or, better, it is just a part of it. More precisely, one obtains the ***LFW*** by using ***VRay***'s ***Frame buffer,*** since it allows one to visualize the image in floating point, in ***HDRI*** for example, contrary to 3ds Max's one. This enables each pixel to contain much more information, beyond the usual 255 ones, and allows one to actually save in a High Dynamic Range Image format. It is possible to save in HDRI from the 3ds Max Frame Buffer as well, but what happens is that one obtains the image in ***LDRI*** (Low **D**ynamic **R**ange **I**mage) anyway, that is the classic .bmp or .jpg .

Until now the way in which 3ds Max shows images has just been modified. But the change of the gamma factor has to affect both textures and 3ds Max's color-selectors. For this reason the two ***Affect Color selector*** and ***Affect Material selector*** options must be activated. This change leads to a better visual correspondence between the colors of the rendering and those visualized, for example, inside the Material Editor. It has to be remarked how the activation of these two options does not affect the render's final result. It just modifies the way 3ds Max is going to visualize colors inside its interface.



■ **Figure 4.142**
By activating gamma influence not only in 3ds Max's Frame buffer but on colors as well, it is possible to obtain a better correspondence between what is observed in the Material Editor and what is to be rendered. The colors of the render are going to be very similar to those visualized in the Material Editor. Instead, colors without gamma correction are going to be too dark inside the Material Editor. The option ***Affect Color Selector*** activates the gamma correction within the color-selector, whereas the option ***Affect material Editor*** affects the color of the spheres and the textures of the Material Editor.

Now that the color-visualization has been made linear, a successive modification within the Gamma and LUT card has to be carried out: the parameter Input Gamma has to be set to 2.2. Most images or textures come from digital cameras, downloaded by Internet or acquired with a scanner. These images usually have a Gamma factor of 1/2.2 already applied. If not they would be very dark if seen within Windows. By using the option Input Gamma = 2.2 the gamma correction is removed from the images. *VRay* is finally going to obtain both textures and colors linearized.

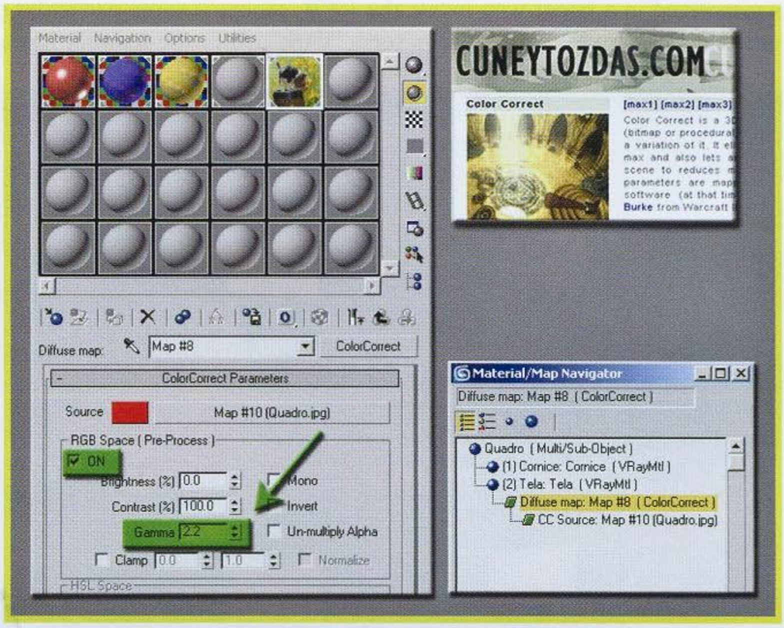


■ **Figure 4.143**
By modifying texture gamma at input a gamma correction is to be applied in order to linearize them. Careful! If the used images have not been adjusted (for example if the sRGB profile has not been assigned at the time of the shot, or they have been saved in RAW format), this value has to be kept 1.0. If not, with input gamma = 2.2, the textures would appear too dark.

Another system for modifying the input of the textures is to overwrite (***Override***) the gamma when one loads it in the Material Editor.

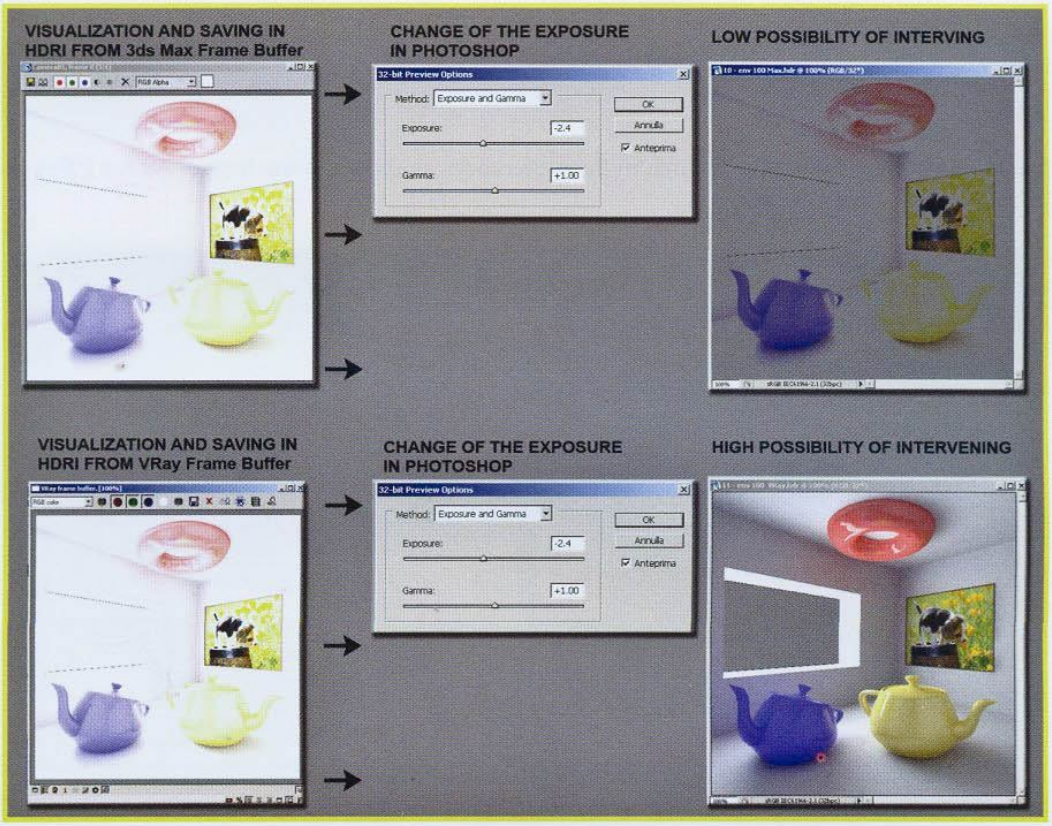
■ **Figure 4.144**
Override: it allows one to change the gamma of single textures. In this case the value in ***Input Gamma*** is not considered.

The last method for changing the gamma of textures is using the **ColorCorrect** freeware plug-in. This software allows one to modify any texture, Bitmap or procedural according to the color, the saturation, gamma contrast, etc… creating in such a way more variations of the same texture. This means it deletes the continuous and boring passages from photo retouch programs, such as Photoshop, to 3ds Max, speeding up its workflow. In the case under examination this plug-in is going to be used for operating on the gamma of the textures. If one uses this program the value of the Input Gamma has to be set to 1.0. in case this happens, the gamma correction would be double, thus producing images which are too dark.
ColorCorrect can be freely downloaded at the address: **http://www.cuneytozdas.com/software/3dsmax/**.



■ **Figure 4.145**
The ***ColorCorrect*** plug-in has been used in the ***Diffuse*** channel, before the Bitmap which is to be inserted. In this case all the changes applied by ***ColorCorrect*** are going to influence the maps it manages.

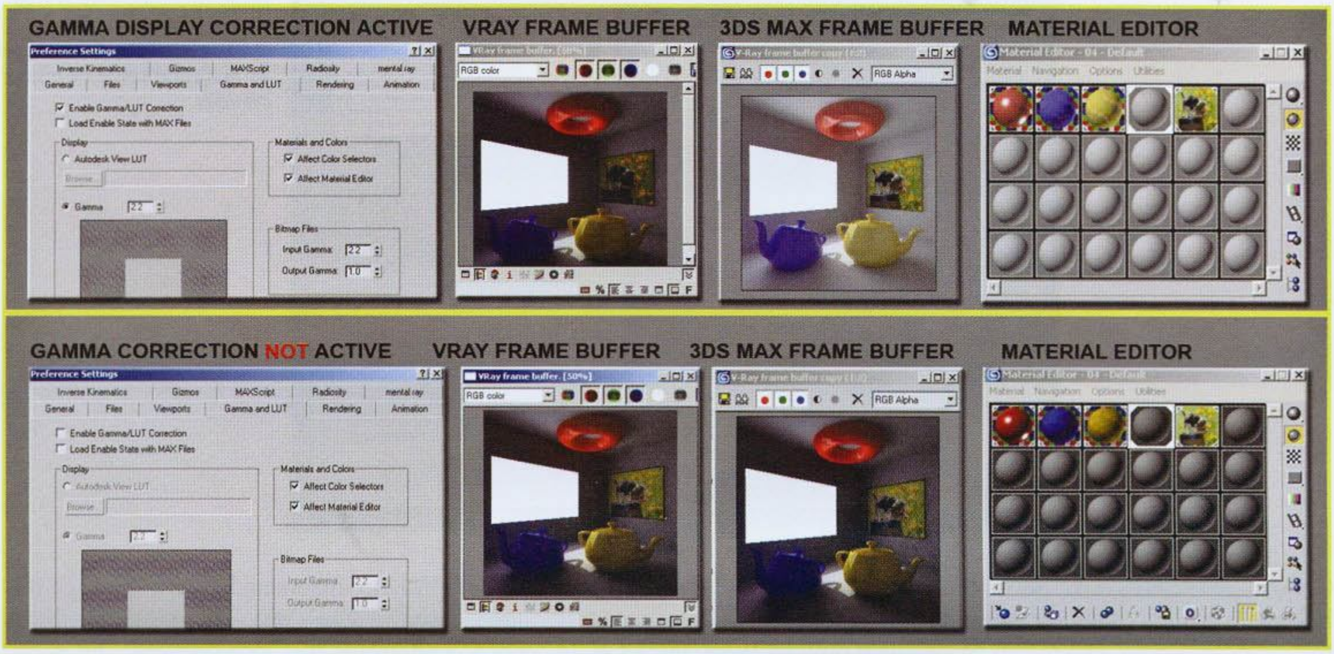
Colors and textures have been corrected. Now it is necessary to operate within ***VRay*** to understand how to manage this information. First of all one has to deactivate the 3ds Max Frame Buffer and use only ***VRay***'s one. The reason for this is that a floating point image can be obtained only by using the ***VFB***. In order to understand this idea properly, this scene is going to be rendered with very high values of the ***Environment*** parameter, so as to produce a particularly overexposed render. First the render is to be computed within the ***VFB***, and then in 3ds Max's Frame Buffer. Both images are to be saved in ***HDRI***. Afterwards they are to be imported in Photoshop. At this time the exposure of the two images has to be changed.



■ **Figure 4.146**
At the end of the rendering VRay creates floating point images which can be visualized throughout their entire range. This phenomenon is visible only inside its ***Frame buffer***.

As can be seen, the images saved in ***HDRI*** from the ***VFB*** maintain all their dynamic gamma. This does not happen by saving from 3ds Max's Frame Buffer. But the ***VFB*** does not consider the Display Gamma 2.2 correction of the visualization activated in 3ds Max preferences.

The render is to be always visualized in a linear way inside the ***VFB***, except if some particular procedures are not going to be carried out. This means that these images will always seem "dark", since the monitor has a ***Gamma of 2.2***.



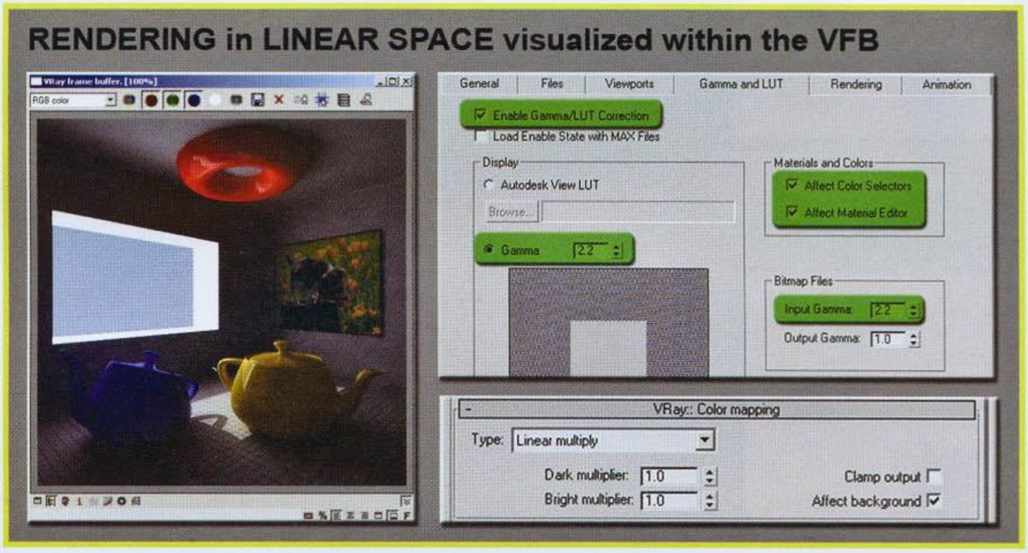
■ **Figure 4.147**
VRay's ***Frame buffer*** does not take into account the 3ds Max gamma applied in the preferences.

To correctly observe the image inside ***VFB*** three systems exist:

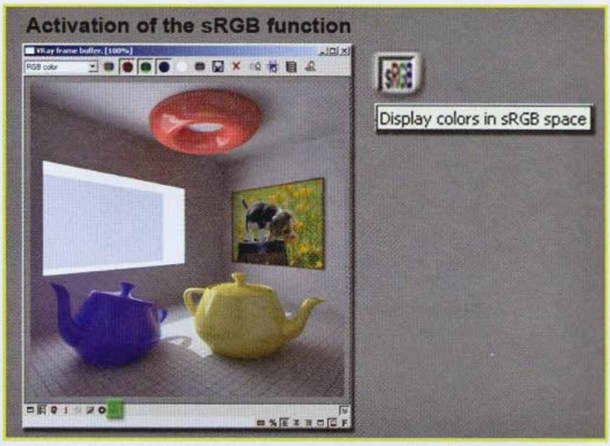
- Using the option ***Display color in sRGB***.
- Using the ***Curve color correction*** present in the ***VFB***.
- Using Color mapping.

In the first two cases one has to proceed by setting the gamma in the preferences as shown in the image. As for ***VRay***, it is necessary to leave the ***Color mapping*** set to ***Linear***.

■ **Figure 4.148**
Settings for the generation of a render in ***linear space***.

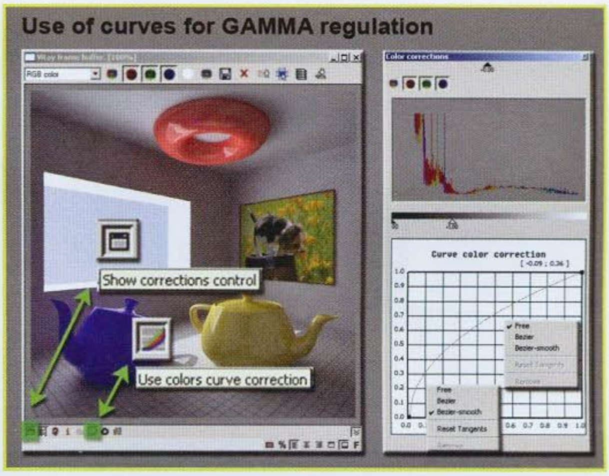


The result is a render which is perfectly in ***Linear space***, but which appears very dark, due to the influence of the Gamma of 2.2 of the monitor, typical of the visualization device. To observe the image in order to perceive it linearly, a gamma neutralising that of the monitor has to be applied, similarly to what happens for digital cameras. The first method requires the activation of the ***sRGB*** option. This way ***VRay*** shows the image assigning the sRGB profile. It is to be remarked that the activation of the sRGB profile only concerns the vision of images within the ***VFB*** and that the image is not going to save this correction when saved.



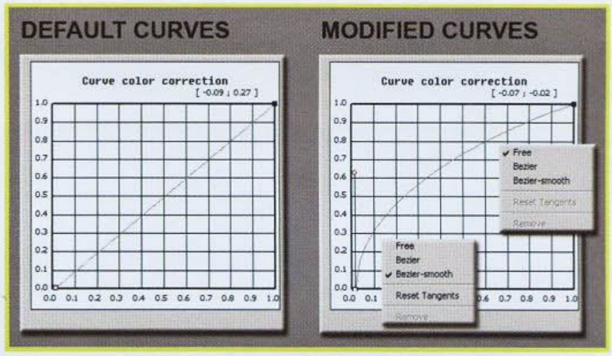
■ **Figure 4.149**
Thanks to the option ***Display colors in sRGB space*** it is possible to modify image visualization so as to allow one to see it correctly.

The second system, concerning ***Curve color correction***, is a less accurate method, but it leads to more flexibility when modifying images. The use of curves can be activated by using the apposite options present in the ***VFB***.



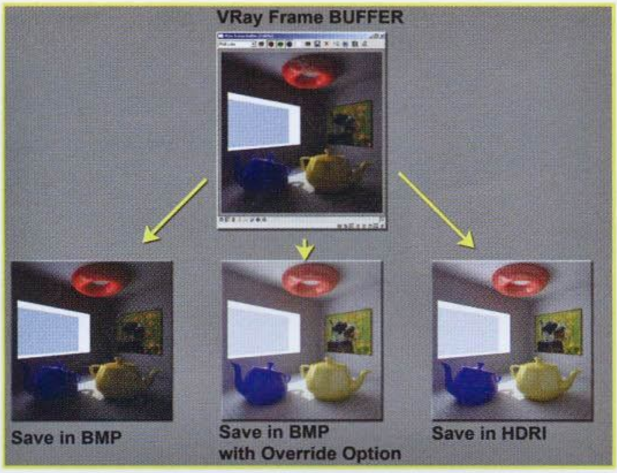
■ **Figure 4.150**
Change of the gamma using the tool ***Color corrections***.

The option ***Show correction control*** allows one to open the panel containing the tools for the color-control. The parameter ***Use colors curve correction*** applies the setting applied within the control panel for color-correction to the image. It is clearly visible how the curve has been modified changing from linear to the opposite of the monitor gamma. To achieve this the starting and final tangents to the curve have been modified.



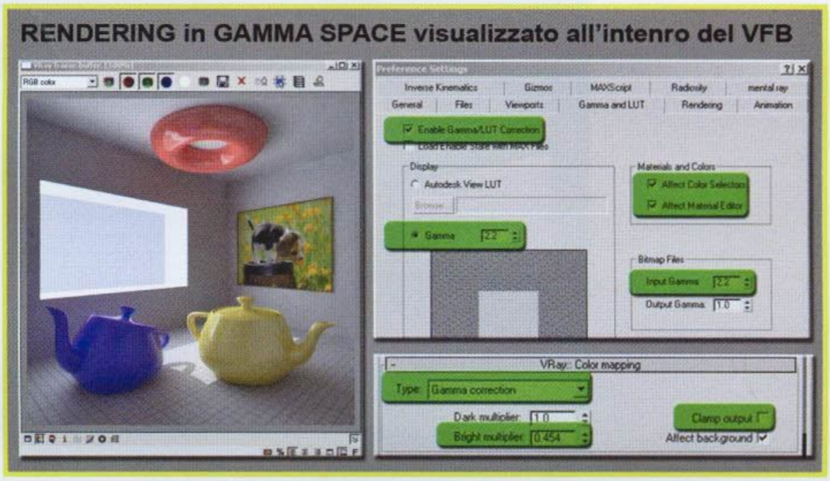
■ **Figure 4.151**
Gamma correction achieved by modifying the curve tangents.

The first tangent, down on the left, has been obtained forcing it to be parallel to the y-axis of the diagram, with length 0.6/0.7. The opposite tangent has instead been modified in ***Free*** mode. The user has a wide freedom in managing correction, with curves and the tangents, for he can modify the curves in real-time and the tangents and, as a consequence, the luminosity/contrast of the render.
The last thing to do is save the render which has just been made, this time from the ***VFB*** and not from the 3ds Max Frame Buffer.
As said before, if one saves in .bmp or .jpg format, the corrections defined through the tool ***Color correction*** are going to reflect in the saved image too. Another way is using the Gamma Override option during the saving of the linear image. But this last system is not advisable, because it is inaccurate. Finally, it is possible to save directly in HDRI format. This is the best solution, because the Gamma 2.2 is to be automatically included in the file and the image is going to have the maximum range possible. A clarification: if the HDRI is opened outside Photoshop, with a program not compatible with the ICC profiles, it appears dark.



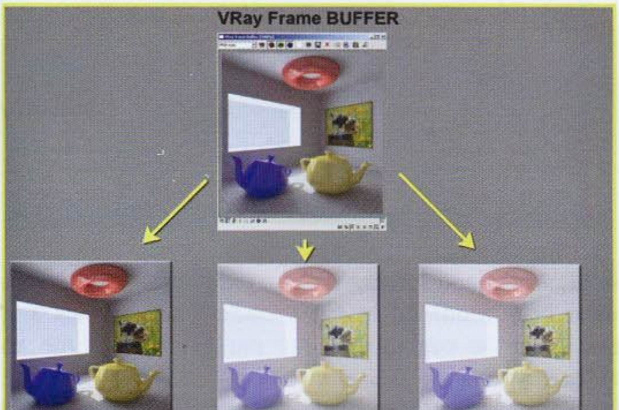
■ **Figure 4.152**
Savings of the VRay's ***Frame buffer*** and vision of it inside Photoshop.

The next correction system of the gamma consists in modifying the ***Color mapping*** mode. Instead of using the ***Linear multiply*** method, the ***Gamma correction*** system is used in this case.



■ **Figure 4.153**
Employment of the ***Color mapping Gamma correction***.

There are two main differences relative to the previous system. The first is the employment of the ***Color Mapping Gamma correction***. The second is the ***Bright multiplier*** parameter is set to 0.454. Actually this value, which is the inverse of the gamma, that is 1/2.2, should not be called ***Bright multiplier***, but rather ***Inverse gamma*** (right wording in v1.5).
At this point, magically, the *VFB* shows a correct image of the render. this happens because the gamma correction has not been applied in post-production by means of the curves, but directly on the pixels of the image. Now one just has to save the render.



■ **Figure 4.154**
Saving of the rendering from the ***VRay Frame buffer*** by using the ***Color Mapping Gamma correction***.

The saving in .bmp or .jpg is enough for obtaining a correct image. Instead using the ***Override*** option would be a mistake, since the gamma correction would be applied twice: once through the ***Color mapping*** and the once again through the ***Override*** option. Same thing for the saving in HDRI format.

Summarizing, the ***LWF*** can be described in 8 main passages:

- Calibrate the monitor, by using the standard value 2.2 for the gamma. This operation is independent of the employment of 3ds Max, *VRay* or any other program, but it might considerably improve the quality of one's work.
- The real ***LWF*** starts from now on. First of all the gamma of 3ds Max is to be set to 2.2 through the preferences. Activate the two ***Affect Color Selector*** and ***Affect Material Editor*** options.
- Set the ***Bitmap Input*** parameter to 2.2. This passage is necessary if one wants 3ds Max to automatically change the gamma of the textures used. 99% of the textures usually employed has an inverse correction to the gamma. The manual change through the ***Color correct*** plug-in would be extremely uncomfortable, as would be the employment of the ***Override*** option as well.
- Set the ***Color mapping*** with the ***Gamma correction*** method, by using ***Dark multiplier*** =1 and ***Bright multiplier*** = 0.454. In case one does not use ***Color mapping***, use the curves available in the ***Frame buffer*** of *VRay*.
- For saving, use the ***LDRI*** format (.bmp, .jpg etc...) in case of employment of the ***Gamma correction***. Without the ***Color mapping*** one has to save by using the ***Override*** option, save in ***LDRI*** by using the curves or in ***HDRI***.

Clamp output - this option "clamps" the colors present in the render for those values exceeding the threshold RGB[255, 255, 255]. What happens when one does not use the ***clamping***?

VRay renders the images in full brightness. This means it is possible to have, in certain circumstances, luminosity values greater than RGB = [255, 255, 255]or float = [1, 1, 1] (the RGB value 255 can be expressed in unitary terms as well), as for ***HDRI*** images. Despite it is not possible to observe these colors on the monitor, this information is present in any case within each pixel. Sometimes it might happen that zones in the shadow, having value float = [0.2, 0.5, 0.5], are adjacent to zones having value float = [5, 5, 10]. Even though a color so bright is represented on the monitor with the white RGB = [255, 255, 255] since it is the maximum reproducible value, in truth it is much brighter.

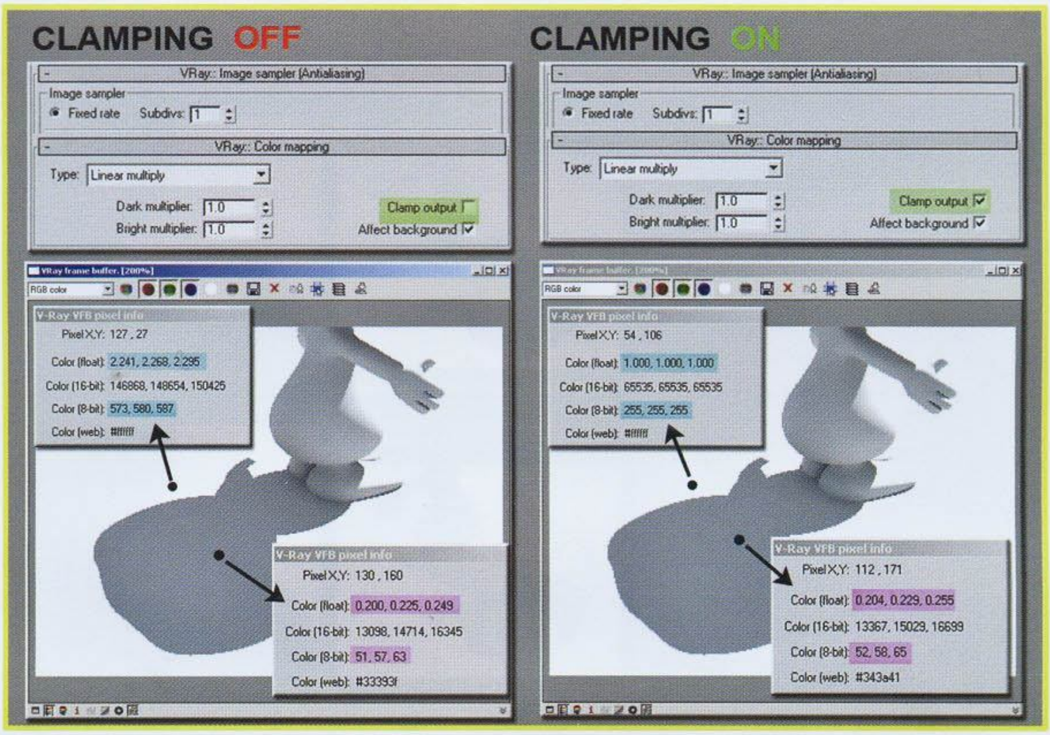
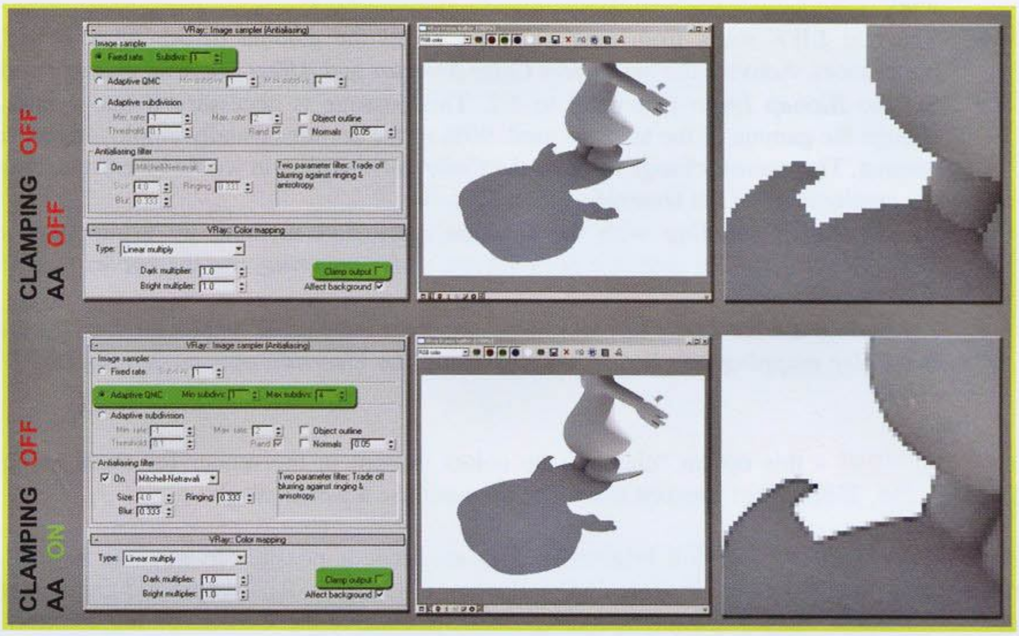


Figure 4.155
Two different renderings accomplished with the ***Clamp output*** option, firstly deactivated and then activated. The two images are almost the same "on the monitor", but it does not the information they carry.

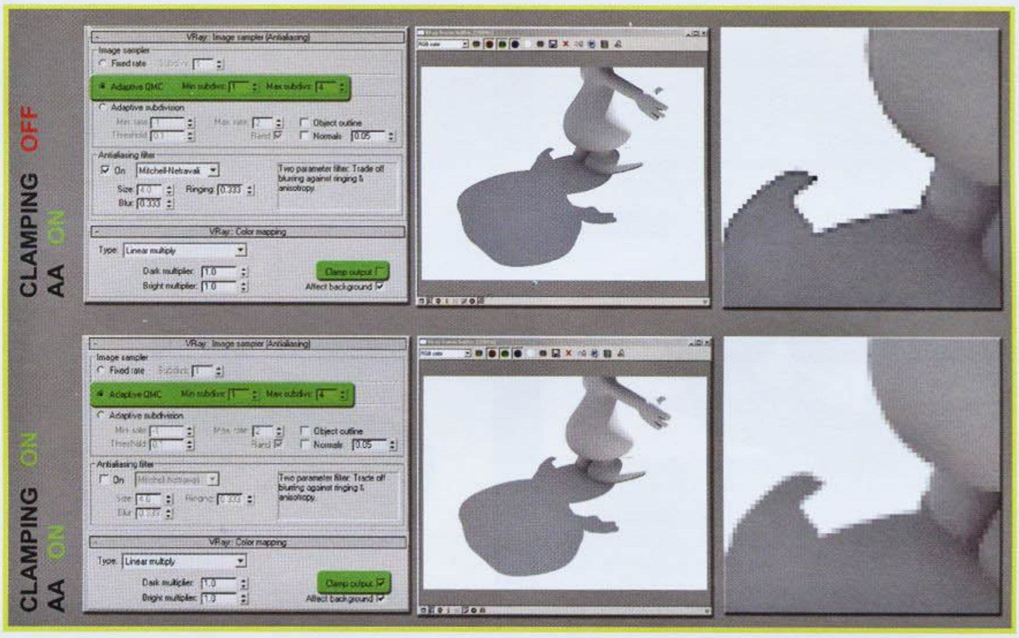
By using *VRay*'s ***Pixel Info*** tool it can be clearly seen how the ***Clamp output*** makes the values which were before higher than float>1.0, become equal to float = 1.0. The value float = [2.241, 2.268, 2.295] has been reduced to float=[1,1,1], while the color 8bit = [573, 580, 587] has become 8bit = [255, 255, 255]. Instead, the colors within the dark area have not been affected by the ***Clamping***, since they are not "over the threshold".

For computing ***antialiasing*** it is necessary to fulfil the average among adjacent pixels. In this case, with float values 5 and 0.5 for example, the result is 2.75, which is a color set much higher than 1(or 255). As a consequence, those areas which need ***antialiasing*** appear as if no operation of ***antialiasing*** has been carried out, and pixels have values higher than 1.0 in any case, with the continuous presence of aliasing.



■ **Figure 4.156**
When treating well-lit areas the ***antialiasing*** creates an effect not directly detectable in the ***VRay Frame buffer***. ***Antialiasing*** is not one of the best.

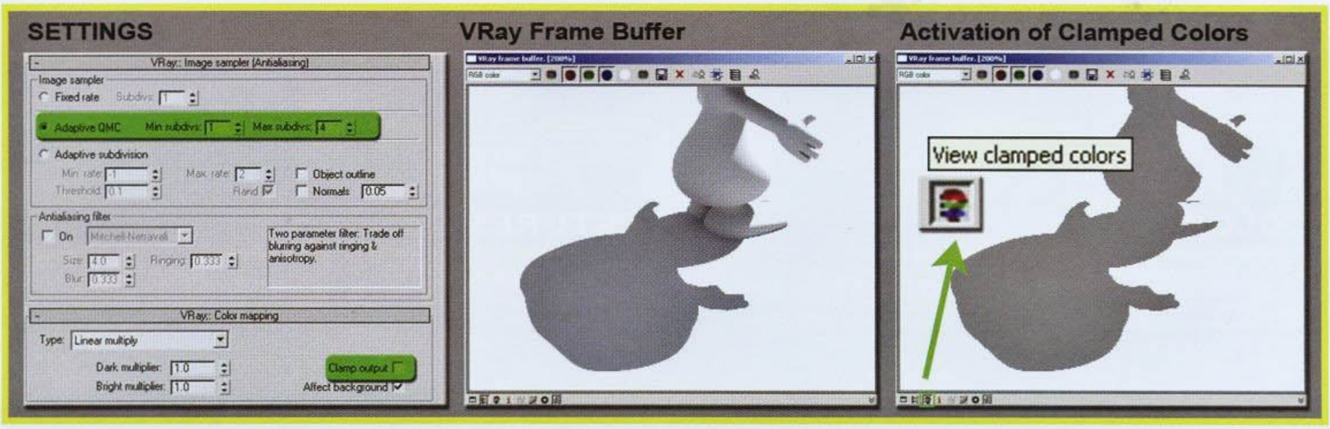
By activating ***Clamping*** one forces *VRay* to consider value 5 and all the values higher than 1.0 as pure white, that is RGB = 255. This time, at the moment of ***antialiasing*** computation, *VRay* has 2 pixels available, of values 1.0 and 0.5. This time ***antialiasing***, equal to 0.75, is correct and visible on the monitor.



■ **Figure 4.157**
Antialiasing is correct by activating the ***Clamp***.

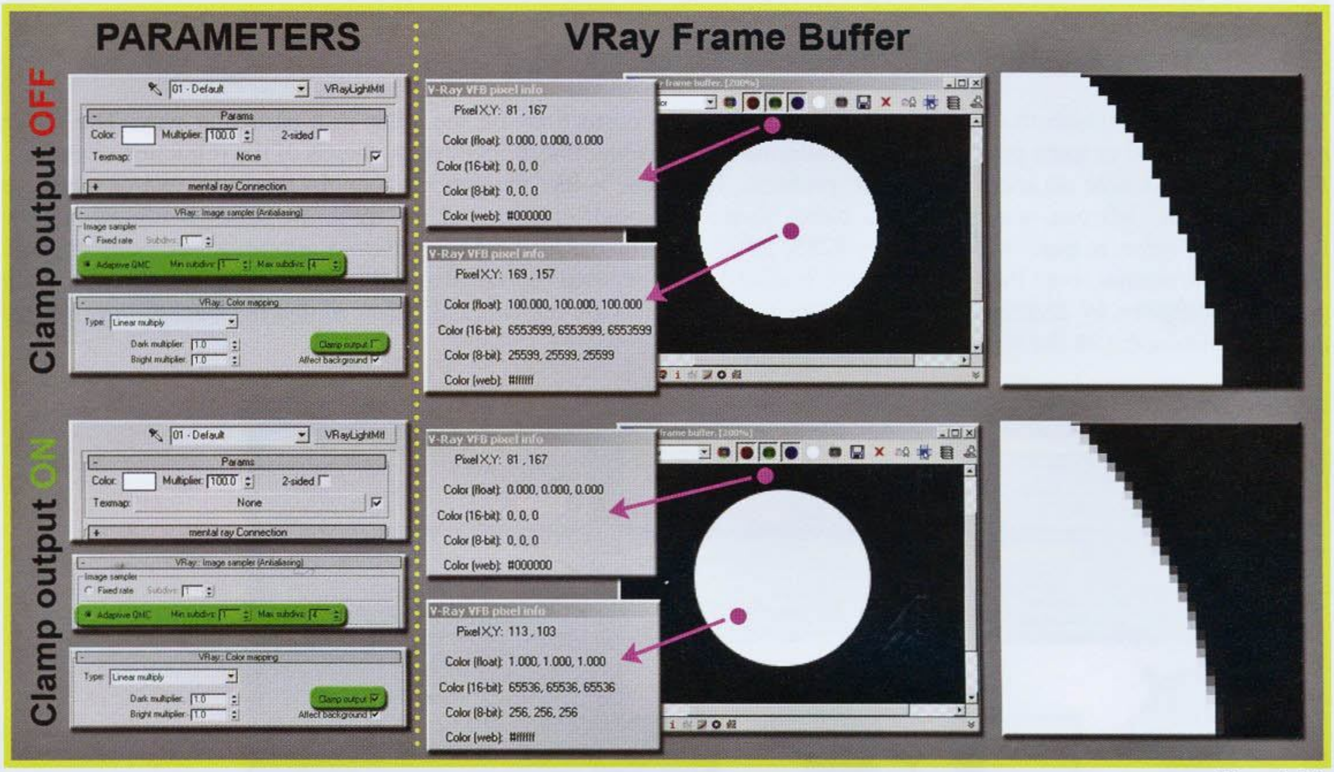
Therefore it might appear logic to keep ***Clamp output*** always activated. It all depends on the use made of the render. In many situations, where post-production is needed, it might be very useful in ***Color correction*** operations to have renders in ***HDRI***, or in any case in floating point. Thus having some colors "cut away" would drastically reduce the possibility of operating.

Moreover, the ***VFB*** allows one to observe the "clamped" colors. It is enough to click on the icon ***View clamped color***.



■ **Figure 4.158**
Visualization of clamped colors through the ***VFB***.

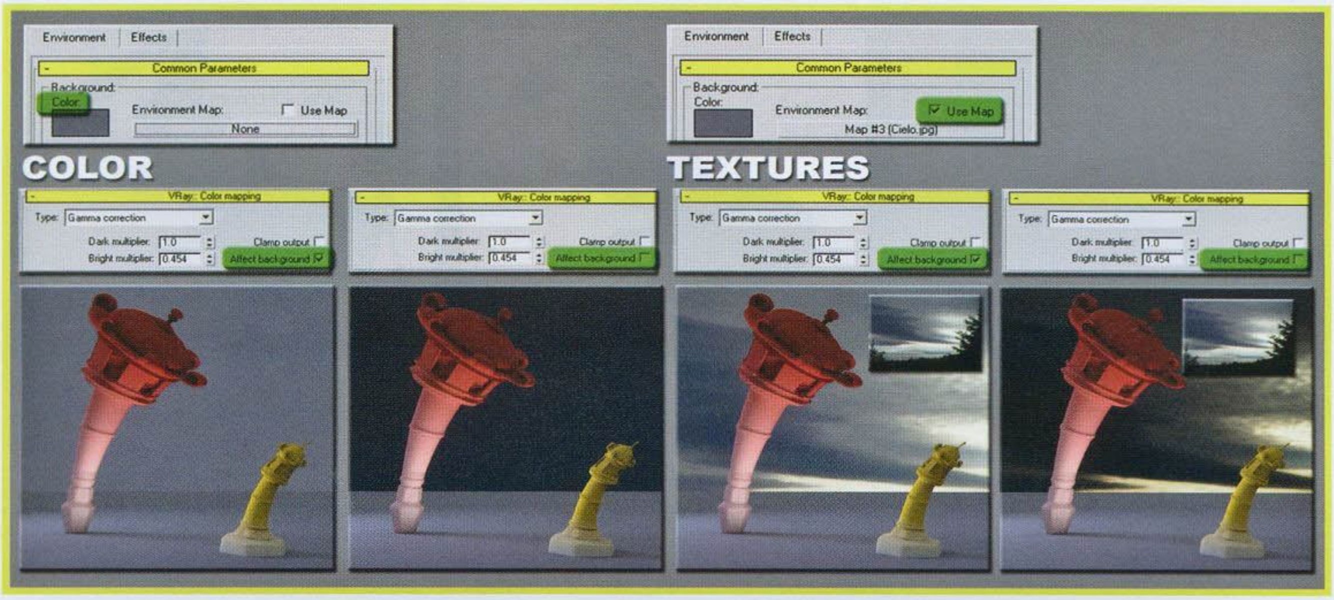
Another example showing how the ***Clamp output*** works can be observed through a geometrical sphere to which a ***VRayLight*** shader has been applied.



■ **Figure 4.159**
Luminous sphere rendered with ***antialiasing*** always turned on, but with different ***Clamp output*** settings.

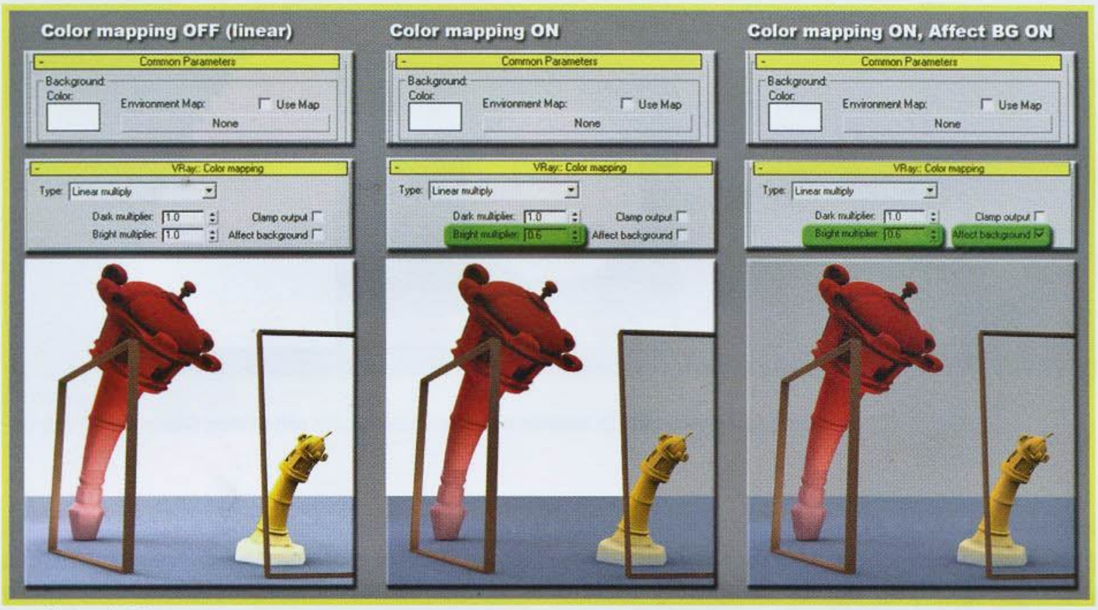
The sphere is very luminous and, if the ***Clamp output*** is active, ***VRay*** interprets this luminosity float = [100, 100, 100]. Although ***antialiasing*** is always active, the sphere is nevertheless "jagged" because of the big difference between completely black float pixels = [0, 0, 0], and the completely white ones in float number format = [100, 100, 100]. To make a correct ***antialiasing*** possible on the sphere, it is necessary to activate output ***Clamping***. Unfortunately it is impossible to obtain images in floating point (that is with ***Clamp output*** deactivated) and with a correct ***antialiasing***, if inside them pixels with RGB values higher than 1.0 are to be found.

Affect background - this parameter activates or deactivates the influence of ***Color mapping*** on color and on background images. If activated, ***Color mapping*** settings involve the background as well. In the opposite case, both colors and the background textures are not affected.



■ **Figure 4.160**
The influence of the parameter ***Affect background*** applied to color and the background textures.

There is a case in which the ***Color mapping*** carries out its corrections on the color in a particular way. The ***Color mapping*** may lead to some problems when rendering a transparent object obtained by means of the refraction channel.
Thanks to an example we show exactly the problem. The scene is like the previous one. In this case, two panels have been added: the left one is a transparent plane obtained through opacity and the right one through refraction. The ***Background*** color is pure white RGB = [255, 255, 255]. In the first test any ***Color mapping*** has been applied. Everything is normal, even thought the image is slightly overexposed. So it is decided to remove this problem by using the ***Color mapping***, by decreasing the ***Bright multiplier***. In this case the image is balanced, but the ***Background*** behind the glass, obtained with the refraction, is very dark. Why?



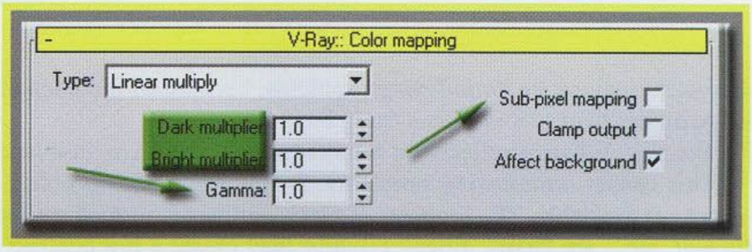
■ **Figure 4.161**
Limitation in the employment of ***Color mapping*** applied to the ***Background***.

Once rendering has been finished, ***Color mapping*** is applied to each pixel. In *VRay* there is no way to apply it to glass, but not to the ***Background*** seen through the panel. Instead, this is possible if the transparency of the shader is obtained through the opacity channel. The same goes for the refractions.

VRAY 1.5 - beta 2

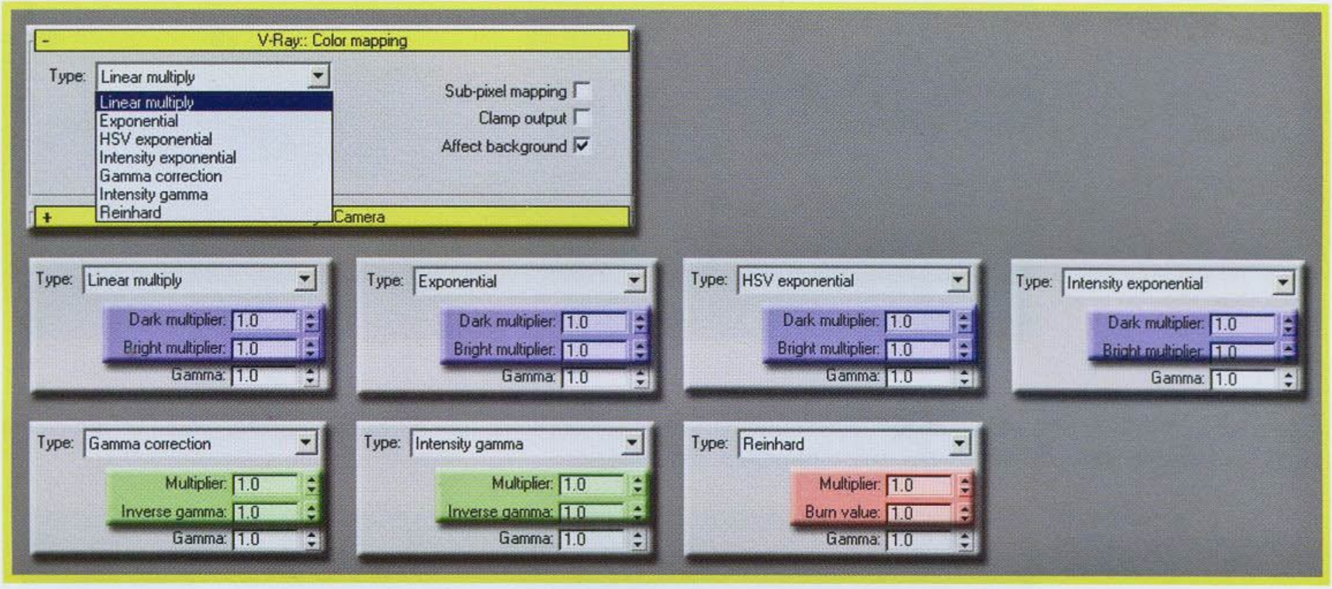
FROM NOW ON EVERY SUBJECT REFERS TO VRay VERSION v1.5 RELEASE CANDIDATE 2. AS THE COMMERCIAL VERSION HAS BEEN RELEASED, AT THE END OF THE BOOK THE NEW FEATURES HAVE BEEN ADDED, AND THE TOPICS WHICH HAVE BEEN MODIFIED OR UPDATED CONCERNING versions v1.4x ARE DISCUSSED.

The introduction of ***VRay v1.5*** brought about lots of new features, changes and improvements, involving the whole ***VRay*** system. Some of these changes are found in the rollout ***VRay: Color mapping***.



■ **Figure 4.162**
The new rollout ***VRay: Color mapping*** in VRay v1.5.

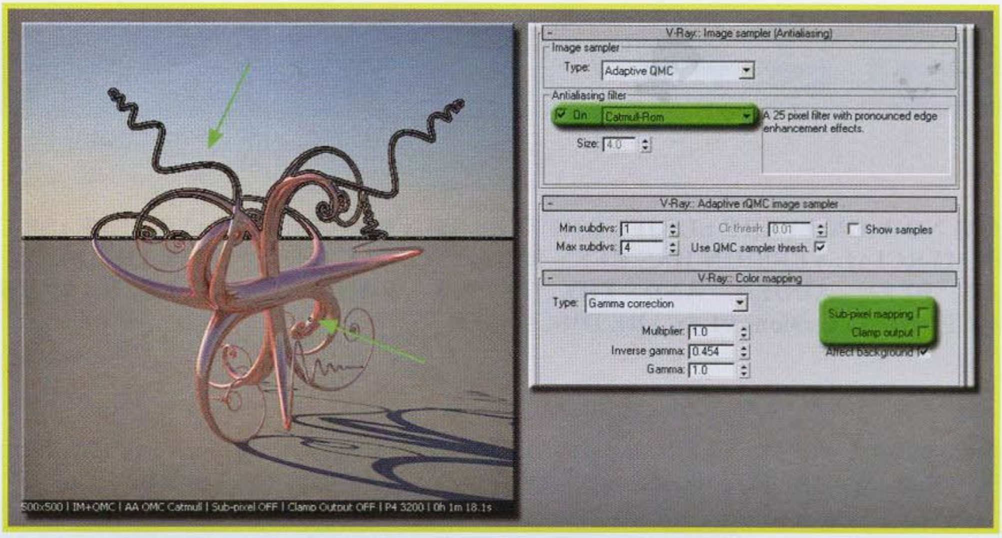
Two new parameters have been introduced: ***Gamma*** and ***Sub-pixel mapping***. Furthermore now by modifying the kind of ***Color mapping***, the names of the multipliers change. Lastly, two new methods have been added: ***Intensity gamma*** and ***Reinhard***.



■ **Figure 4.163**
Various ***Color mapping*** methods.

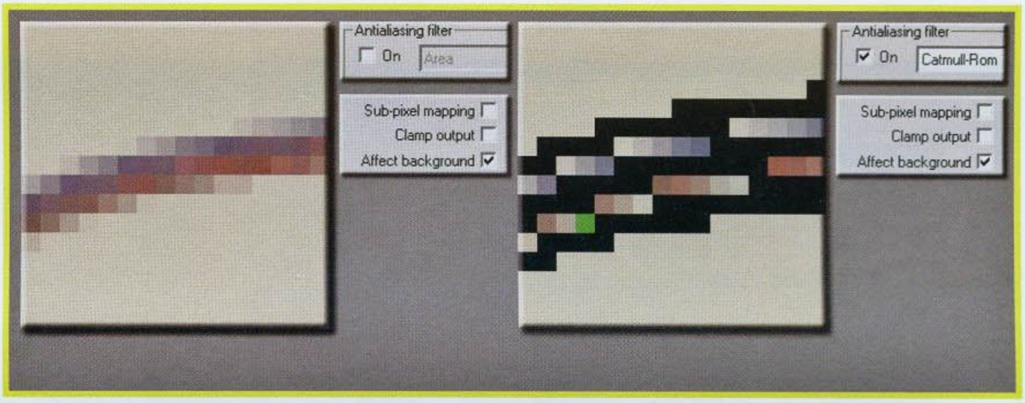
Sub-pixel mapping - this option, which was hidden and active by default in the versions prior to 1.5, checks whether ***Color mapping*** is applied to single pixels or if it is applied at sub-pixel level. In ***v1.5*** it is deactivated by default and this ensures more correct results, applying the ***Color mapping***, in such a way, to the single sub-pixels. But there are some cases where it is particularly necessary to pay attention.

In the next test a classic outdoor scene, illuminated with the new ***VRay v1.5*** system for solar simulation is shown. The scene is seen from the new ***VRayPhysicalCamera***. In actual fact the sun has a high luminous value and this value is precisely reproduced within ***VRay***'s ***Sun/Sky*** system.



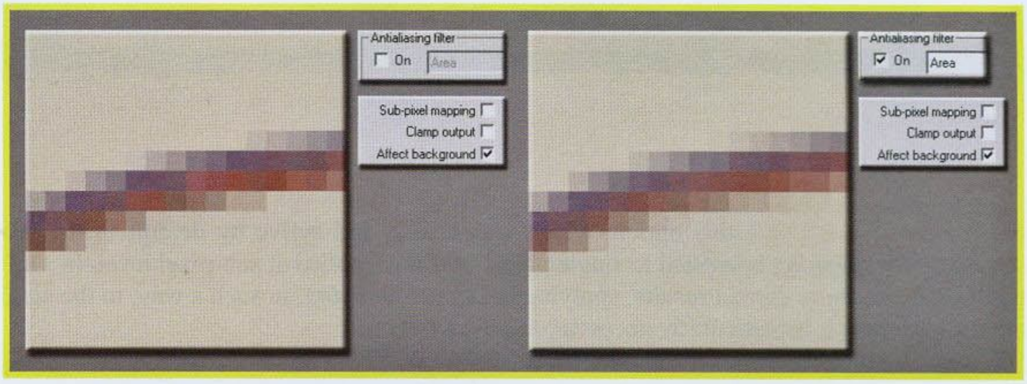
■ **Figure 4.164**
Artefacts due to the high luminosity of the sun.

In some circumstances this system may cause ***antialiasing*** problems, because of the phenomenon previously discussed concerning pixels with a value higher than 1: it is not a system bug!
The sun is a small point in the sky. When it reflects on a metal object, the areas involved may assume a high RGB value, often higher than 1. This causes ***antialiasing*** problems. A solution to this problem might be to make the sun invisible. But this option is not yet available. The problem of black outlines is instead caused by a kind of filter used in ***antialiasing***. ***Catmull***, ***Mitchell***, etc... are "hard" filters. Usually they create a big contrast between areas with higher variation of color and intensity.



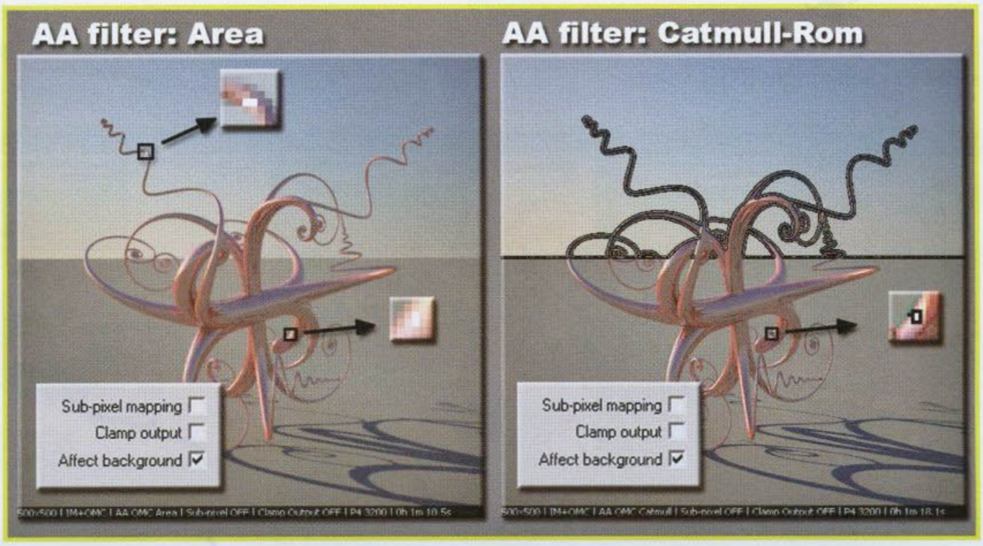
■ **Figure 4.165**
Artefacts due to the use of the ***Catmull*** filter.

A solution to this kind of problem might be the employment of softer filters, like for example ***Area***, which do not create "rings" around the edges. This way the dark outlines are removed.



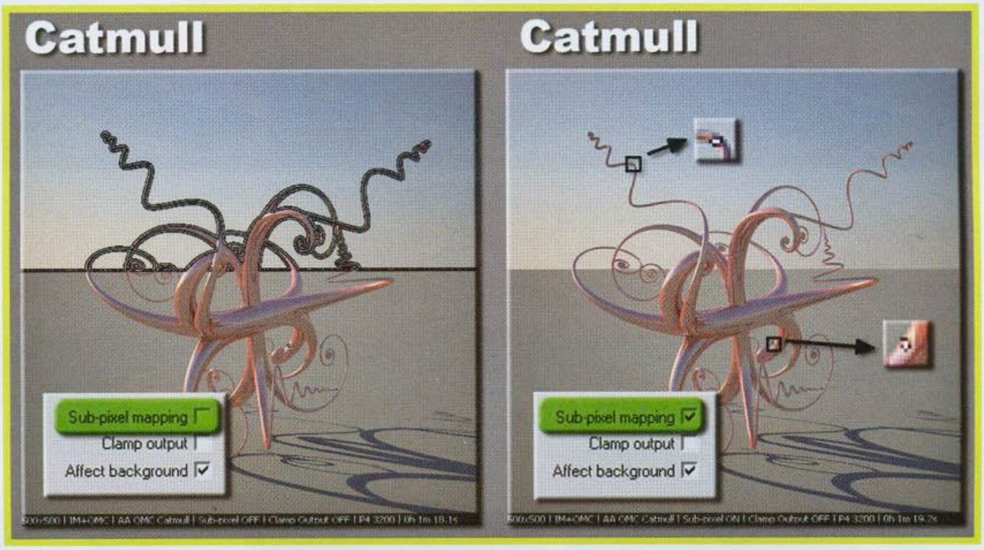
■ **Figure 4.166**
Correct rendering thanks to the ***Area*** filter.

Even with the use of the *Area* filter, or in any case of "soft" filters, some rendering problems can occur in any case.



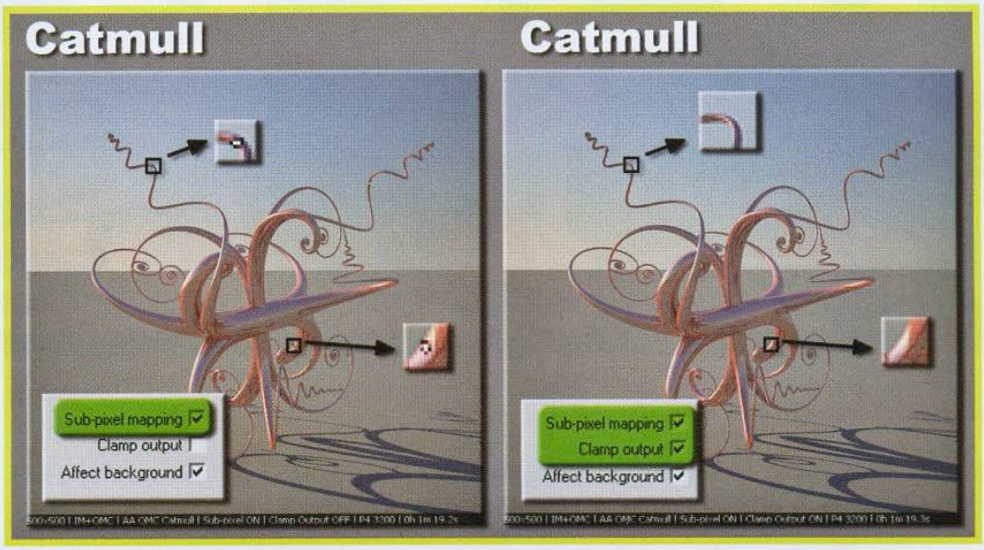
■ **Figure 4.167**
Comparison between the use of ***Area*** and ***Catmull*** filters.

By activating ***Sub-pixel mapping*** option and the hard filters, the result is better, but some small problems persist.



■ **Figure 4.168**
Activation of the ***Sub-pixel mapping*** parameter using the ***Catmull*** filter.

By activating the parameter ***Clamp output*** also, the render is obviously correct.



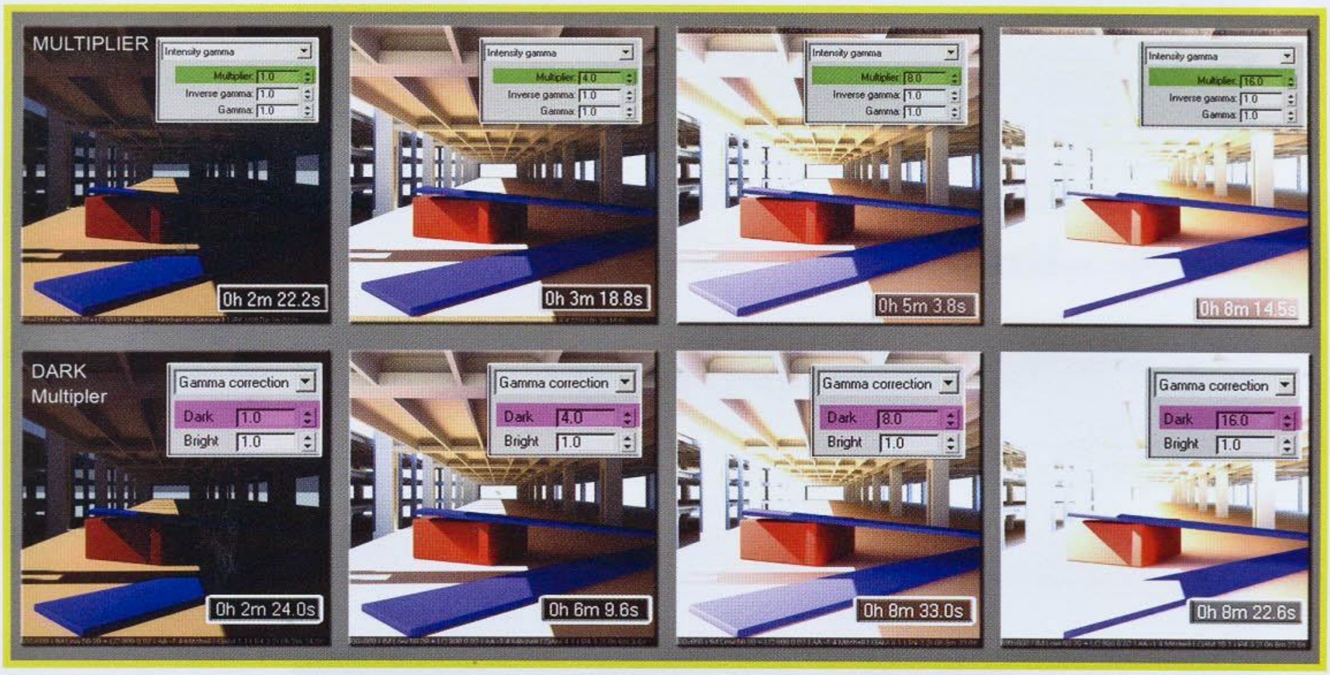
■ **Figure 4.169**
Activation of the parameters ***Sub-pixel mapping*** and ***Clamp output***.

Two new ***Color mapping*** methods have been introduced with the introduction of ***VRay*** version 1.5: ***Intensity gamma*** and ***Reinhard***.

Intensity gamma - like the ***Exponential*** mode, the ***Gamma correction*** also has a method applying the gamma curve to color intensity instead of each single RGB value.

■ **Figure 4.170**
Some examples of ***Color mapping*** using the ***Intensity gamma*** mode.

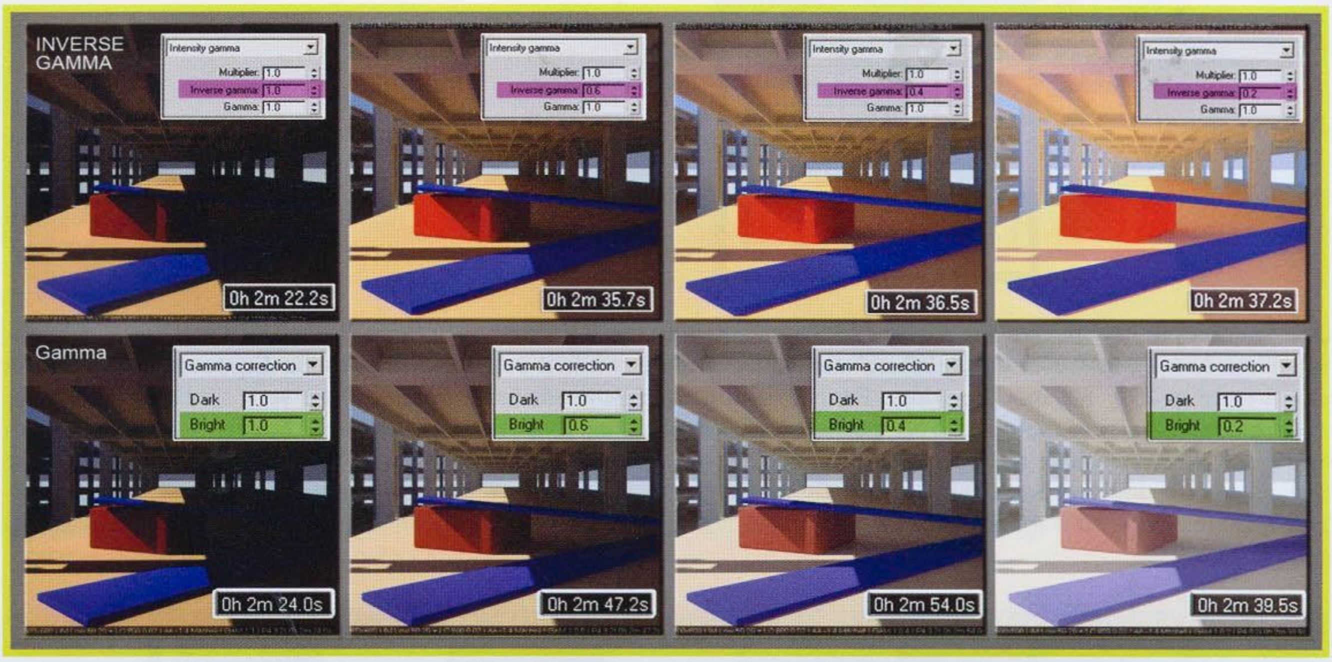
Now the two typologies ***Gamma correction*** and ***Inverse gamma*** are to be compared.



■ **Figure 4.171**
Comparison between ***Gamma correction*** and ***Intensity gamma*** modes.

In both solutions the result is the same just by modifying the ***Multiplier*** (***Dark*** in previous versions to ***VRay v1.5***).
In this case the ***Multiplier*** value works as a general multiplier of the luminous intensity.

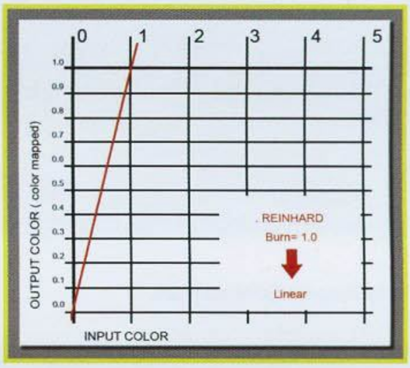
Instead by altering the parameter ***Inverse gmma*** (or ***Bright multiplier*** in previous versions to ***VRay v1.5***) the situation changes drastically.



■ **Figure 4.172**
Comparison between ***Gamma correction*** and ***Intensity gamma*** modes: modification of ***Inverse gamma***.

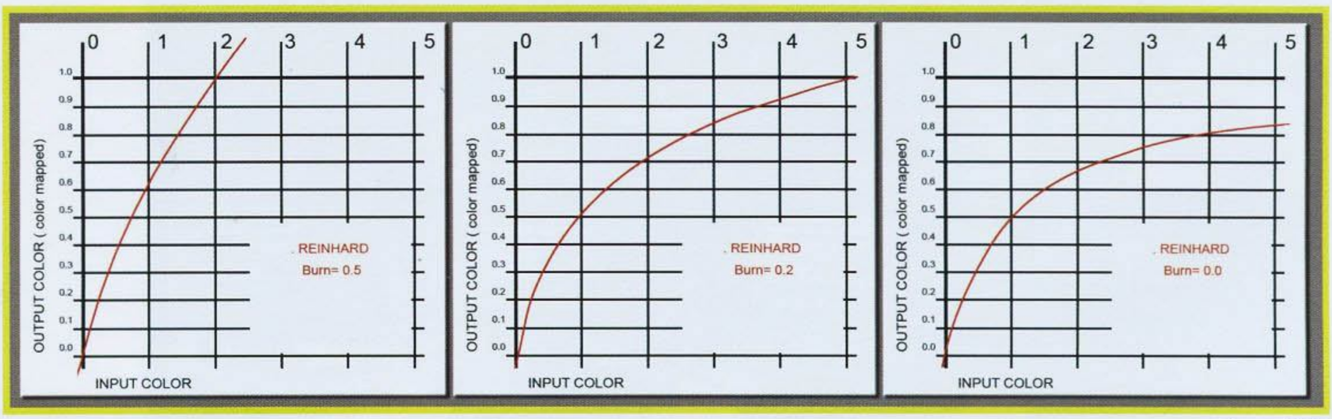
By using the ***Intensity gamma*** method *VRay* tries to preserve the original color of the models, by removing the "faded" effect typical of the ***Gamma correction*** method.

Reinhard - with this method it is possible to obtain a result defined as a mixture of the ***Exponential*** and ***Linear*** methods. For better understanding the functioning of the ***Color mapping Reinhard*** some charts are shown.



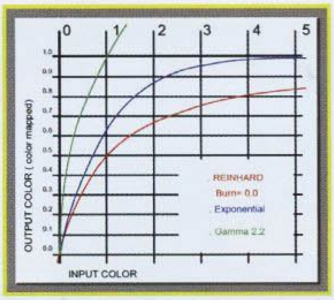
■ **Figure 4.173**
Linear behaviour of the ***Reinhard*** method.

One can deduce from the image above how the default setting leads to a linear behaviour of the color mapping.



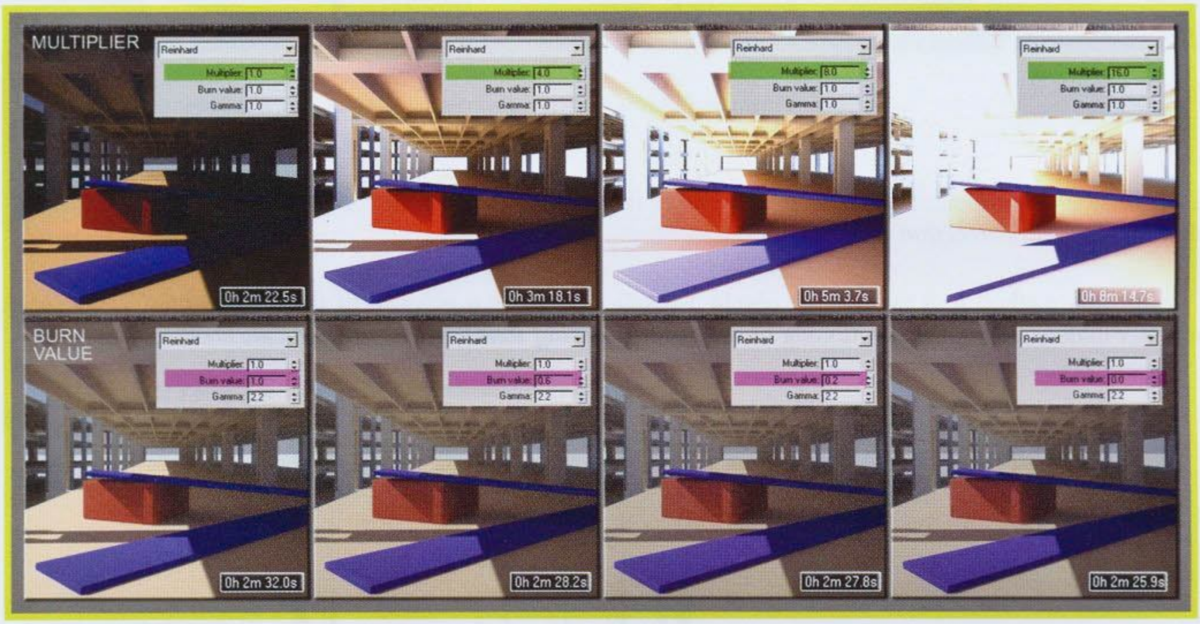
■ **Figure 4.174**
Color correction obtained by modifying the ***Burn value*** parameter.

The graph changes by varying the ***Burn*** parameter.



■ **Figure 4.175**
Comparison between ***Reinhard***, ***Exponential*** and ***Gamma 2.2*** modes.

One can observe how neither the ***Exponential*** mode, neither the ***Reinhard*** one (***Burn value*** 0.0) reach the maximum-white line (y = 1). However the ***Reinhard*** method has a more constant trend compared to the ***Exponential*** one, which grows more quickly towards the white. Furthermore, both ***Exponential*** and ***Reinhard*** are very similar to the linear mode in the first part of the curve. This means these methods preserve dark colors very well. Finally, with ***Burn*** values of 0.5, the ***Reinhard*** curve intersects the white line (y = 1) in x = 2 in other words, with this setting the intensity of the original color is doubled. Same thing for the ***Burn***=0.2. In this case the intensity of the original color is multiplied by 5. This relationship is valid for any other color and allows one to choose precisely the values of the colors one wants to map.



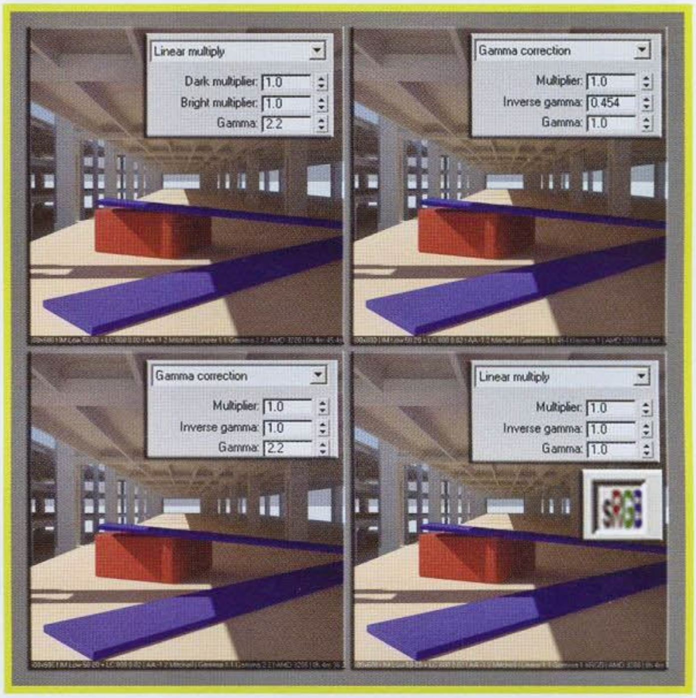
■ **Figure 4.176**
Some examples of ***Color mapping*** by using the ***Reinhard*** mode.

■ **Figure 4.177**
Comparison between ***Reinhard*** and ***Gamma correction*** modes.

As for the previous comparisons, the variation of the ***Multiplier*** value changes the general intensity of the render. Instead, by using different values of ***Burn value*** a slight variation of image saturation can be noticed.

Gamma - this parameter controls the gamma correction of the color. It represents the inverse value of the gamma which is normally used in the ***Inverse gamma*** option in the ***Color mapping Gamma correction***. For example, if one wants to adjust an image for a monitor with a gamma of 2.2, one has to set Gamma = 2.2. By using the following methods, the final result does not change:

. ***Linear color mapping*** - ***Dark mult.*** 1.0 - ***Bright multiplier*** 1.0 - ***Gamma*** 2.2
. ***Gamma correction*** - ***Multiplier*** 1.0 - ***Inverse gamma*** 0.454 - ***Gamma*** 1.0
. ***Gamma correction*** - ***Multilper*** 1.0 - ***Inverse gamma*** 1.0 - ***Gamma*** 2.2
. ***Linear color mapping*** - ***Dark mult.*** 1.0 - ***Bright multiplier*** 1.0 - ***Gamma*** 1.0 - ***VFB sRGB button*** ON



■ **Figure 4.178**
Same renders obtained by using the various settings of the ***Color mapping***.

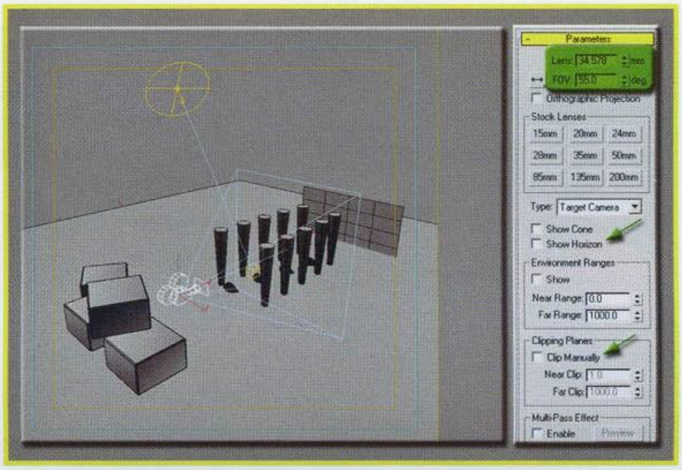
In truth some differences are present. Using gamma correction at the ***Color mapping*** level, instead of the ***Frame buffer*** level only, leads the darker zones to be brighter already during the GI computation, for example. This way *VRay* can sample more easily those areas which would result darker if not "brightened" by ***Color mapping***. Instead, by modifying the gamma only during post-production inside the ***VFB***, the dark zones may not be sampled and have greater noise.

Many types of ***Color mapping*** exist, and there is no best choice among them. It all depends on what one wants to obtain and the purpose of the rendering. Most of the time one wants to obtain very photorealistic and balanced renders. For this purpose the ***LWF*** is the best method. Other times one wants to obtain very contrasted images. Here the best solution is the linear type. All this is possible thanks to *VRay* and its ***Color mapping***.

VRay: Camera

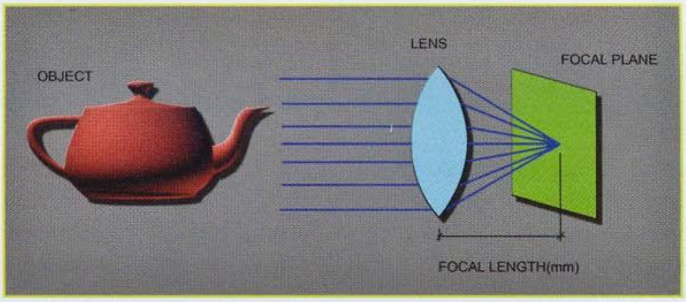
INTRODUCTION

A scene is often observed and rendered through the 3ds Max camera. This object has some parameters allowing one to modify the focal length of the lens, the field of view of the camera, as well as other parameters which are less "photographic" such as the setting of the clipping planes or the possibility of seeing the horizon.



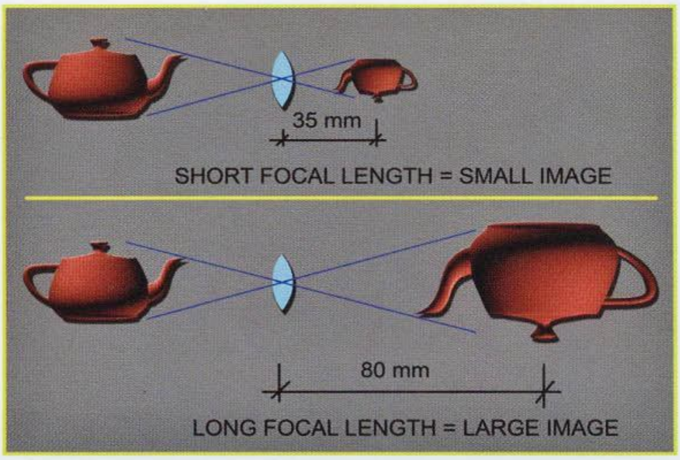
■ **Figure 4.179**
3ds Max cameras with the main photographic parameters.

A converging lens focuses the rays coming from a very far object in a point. The distance between the centre of the lens and the focal plane, that is the plane on which the clear image of the subject is formed, is the focal distance of that lens.

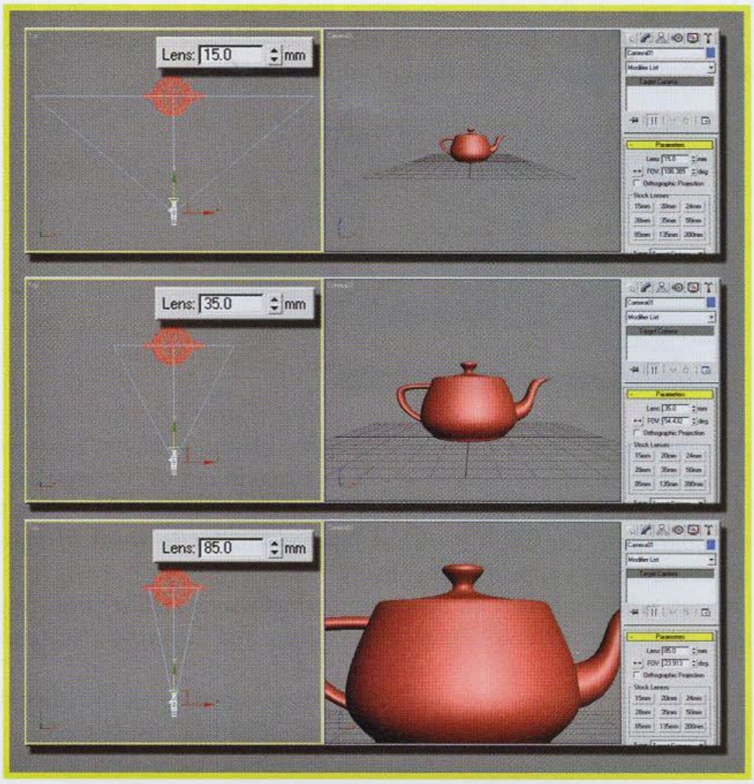


■ **Figure 4.180**
Focal length.

The focal length determines the width of the framed object. This factor has two important consequences. The first is that the distance between the object and the lens being equal, a lens with long focal distance, such as telephoto lens, produces an image bigger than a lens with short focal distance like the wide-angle lens.



■ **Figure 4.181**
Different focal lengths.

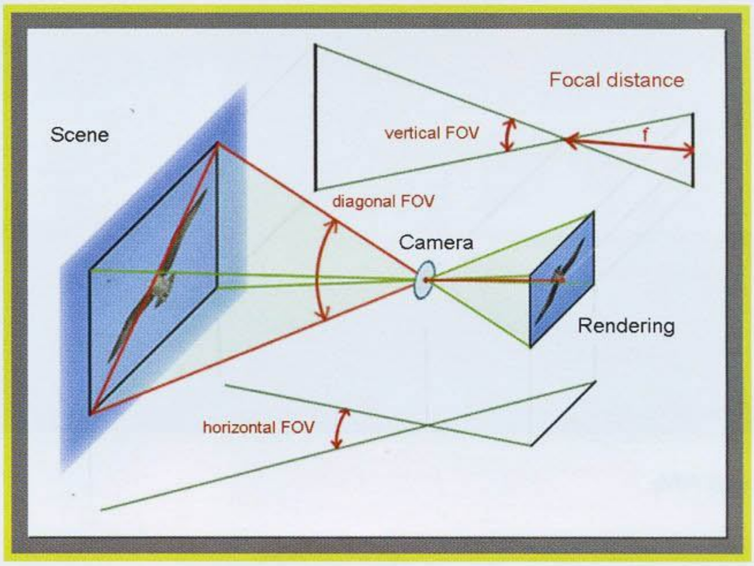


■ **Figure 4.182**
Change in the focal length within 3ds Max.

Inversely, in order to keep the size of the image on the film constant, if the focal length of the length in use is changed, one necessarily has to change the distance from which the images are shot, moving further away the higher the focal length used is. Usually the focal length of a lens is carved on the frame.

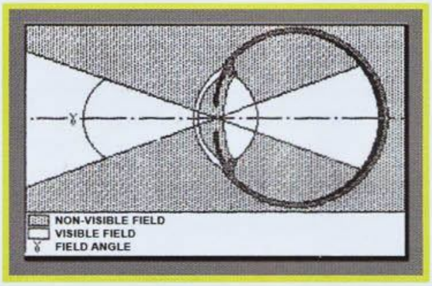
■ **Figure 4.183**
A telephoto lens. The 50 mm lens is the closest lens to the human eye.

Another parameter exists, also connected to focal length, called ***FOV***, that is ***F***ield ***O***f ***V***iew.



■ **Figure 4.184**
Schemes of the ***FOV*** values and the focal distance.

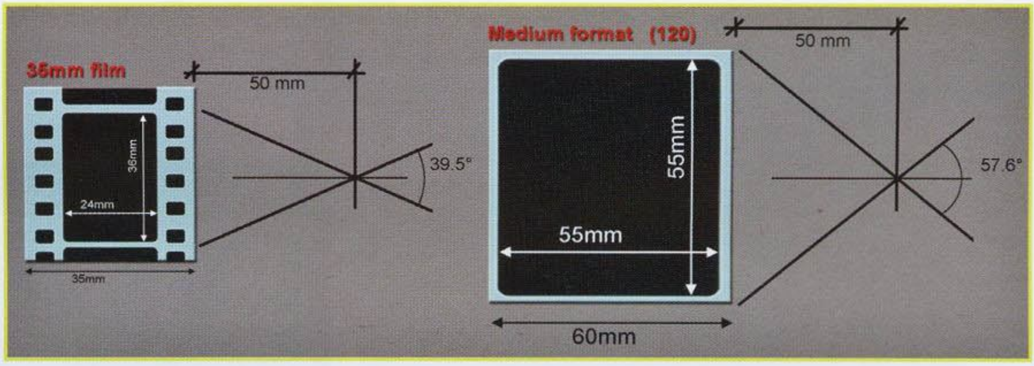
In the human eye the field of view is the maximum angle inside which the external one can observe the external environment. Practically this means that the human eye is able to see the world at 360°, but it is able to see it only bit by bit; this visible portion (measured in degrees) is the field of view (FOV).



■ **Figure 4.185**
Field angle of the human eye. The lens having the most similar FOV to the human eye is considered as normal.

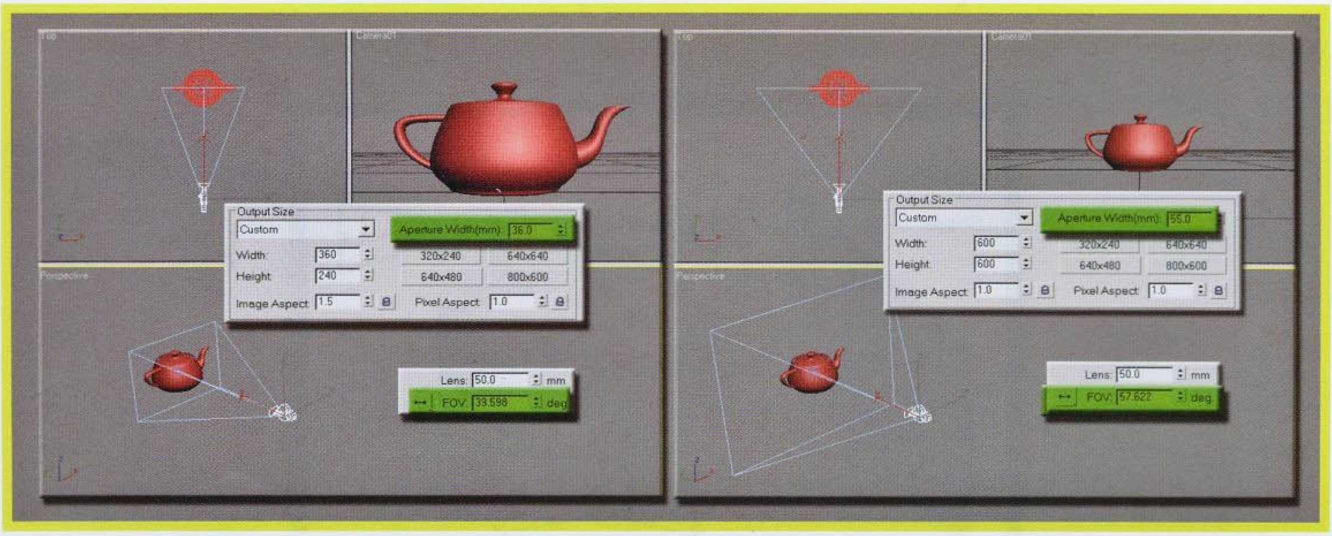
The field angle can change if the eye is still or it moves, but from the photographer's point of view the field angle can be set at 45 degrees, or in any case from a minimum of 40 to a maximum of 55 degrees. An instrument observing a landscape with wider field of view than the eye, must allow a portion of space wider than the normal one to fit inside the portion visible by the human eye. Therefore it makes images smaller in order to make them fit in the available space. As a consequence observed objects appear further away.
The exact opposite effect occurs when one uses an instrument which has a smaller field angle compared to the human eye: objects must be enlarged in order to cover all the available space and therefore seem closer.

For each film format, the focal length of a lens sets its field angle. It is important to notice that the field angle of a lens does not just depend on the focal length, but on the size of the film it has to cover as well.



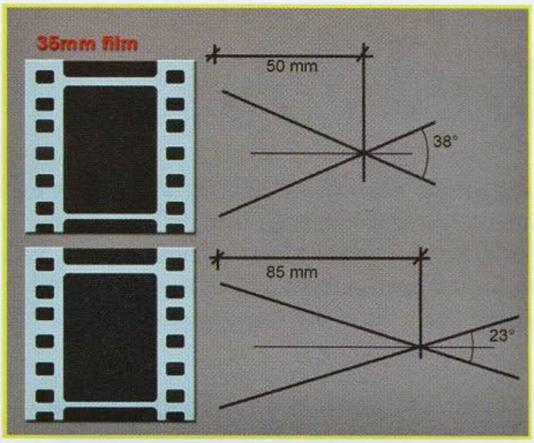
■ **Figure 4.186**
Variation of the ***FOV*** due to the employment of different photographic films. The long side of the frame of the 35mm films is 36x24mm, while for the 70mm one it is 48x36mm.

In 3ds Max the length of the film can be simulated thanks to the parameter ***Aperture Width*** found in the section ***Output Size.***



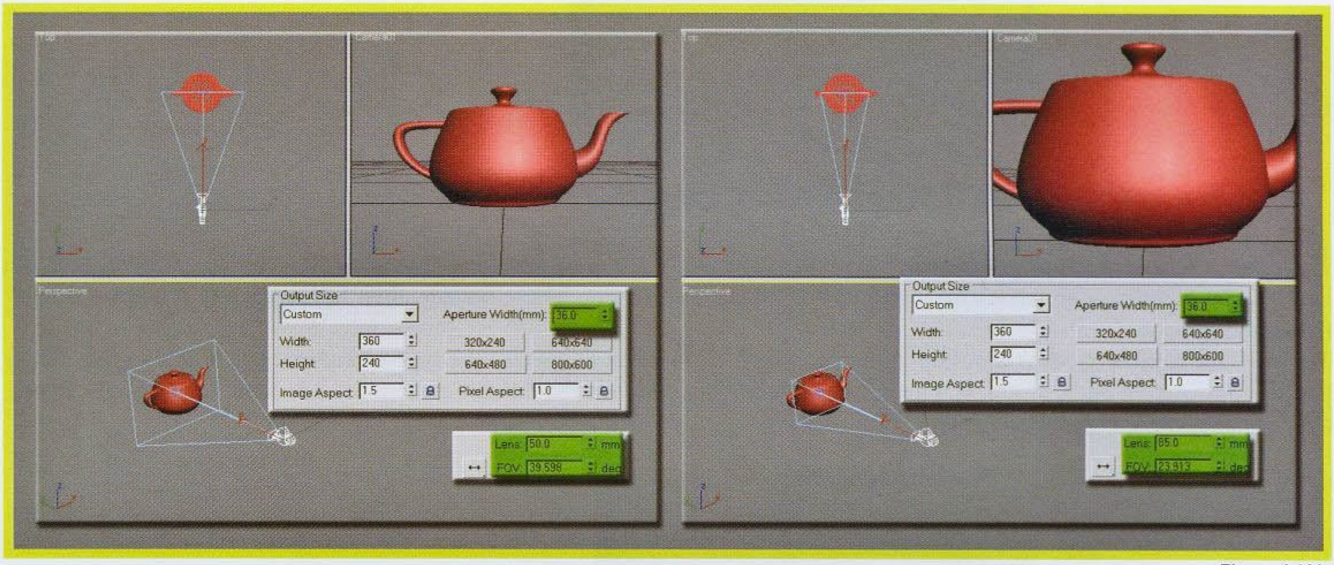
■ **Figure 4.187**
Different films generate different width ***FOVs***.

It is clearly visible how the focal distance of 50mm being equal, it is possible to enlarge the vision angle with a wider film, rendering a larger part of the scene.
As a consequence, with an equal format, a lens with longer focal distance is characterized by a more narrow field angle.



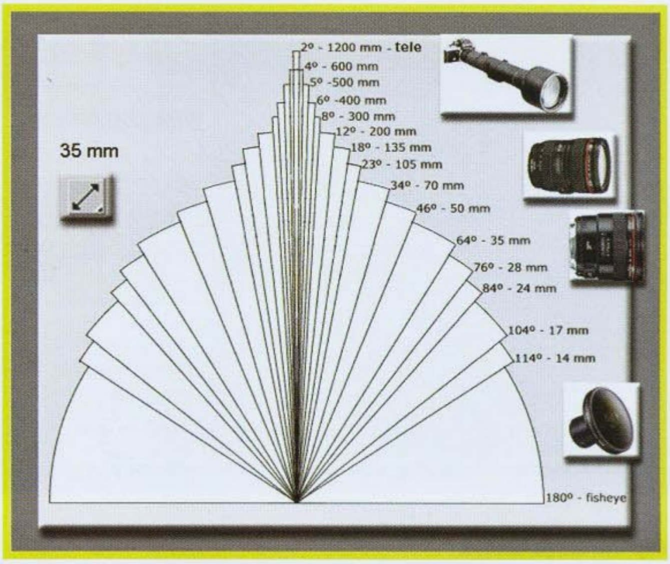
■ **Figure 4.188**
The film being equal, the field angle varies with the focal length.

In 3ds Max this effect can be simulated by using the parameter ***Lens*** available in the standard camera.



■ **Figure 4.189**
A wide-angle lens has a big ***FOV***.

The following illustration shows the angles of different focal lengths for the 35mm format. It should be noticed that ***FOV*** degrees are calculated diagonally and not horizontally.

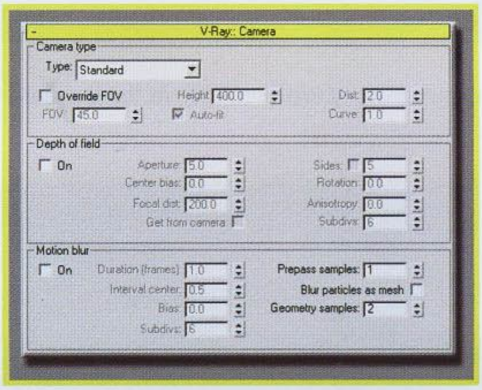


■ **Figure 4.190**
Summarizing table.

PARAMETERS

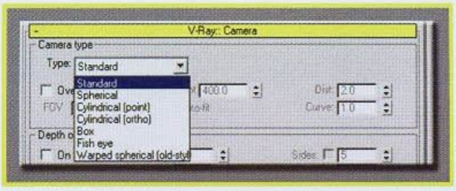
Camera type

Until now we have observed all the parameters that 3ds Max offers for the management of cameras. As well as being able to render via the standard 3ds Max cameras, using all its parameters, ***VRay*** offers new parameters which are not contained in 3ds Max. Everything is controlled by a single rollout, ***VRay: Camera*** which contains, as well as typical camera functions, a section for the management of Depth-of-field (***DOF***) and one for ***Motion blur***. With the parameters in the ***Camera type*** section one can control the way geometry objects are projected within the camera and consequently rendered.



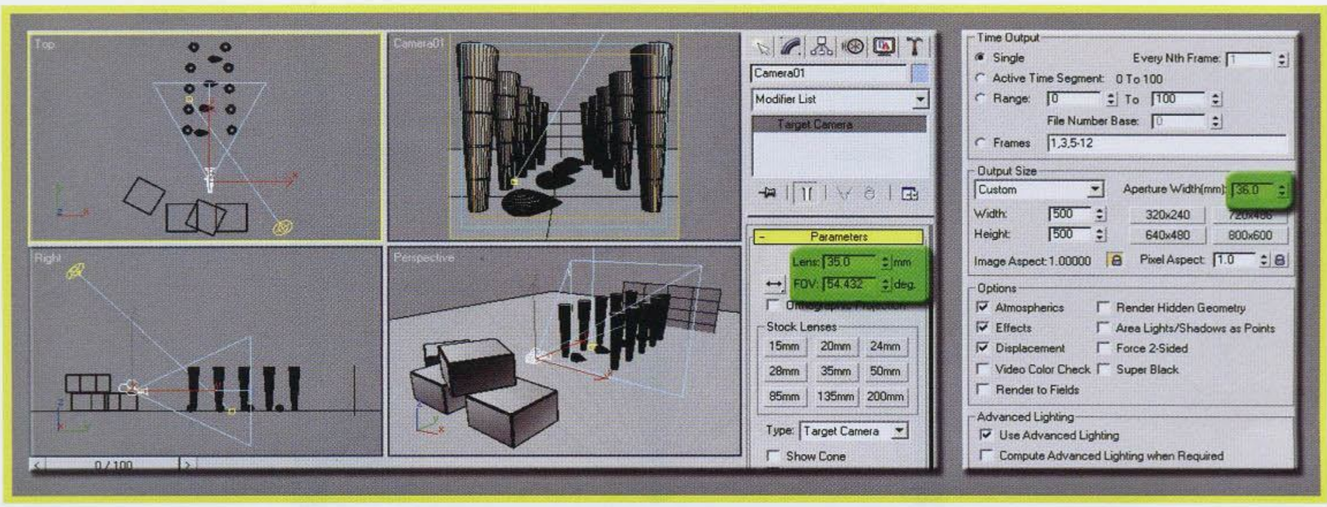
■ **Figure 4.191**
Management panel for cameras in VRay.

Their use is simple. First of all one starts with the creation of a classic 3ds Max camera. After, all the parameters of the case are set and then one passes on to the options that ***VRay*** offers. These options then influence any renders made by the active camera.
The first thing one notices is a pulldown menu, which allows one to specify the type of camera one wants to use. ***VRay*** has seven available ones, amongst which the standard one, which is the default camera for 3ds Max, a spherical camera, cylindrical (***point***), cylindrical (***ortho***), ***Box***, ***Fish eye*** and ***Warped spherical***.



■ **Figure 4.192**
The seven VRay cameras which allow one to observe the digital world.

We will now proceed to analyze each type of camera and suggest examples of renders for all the parameters.

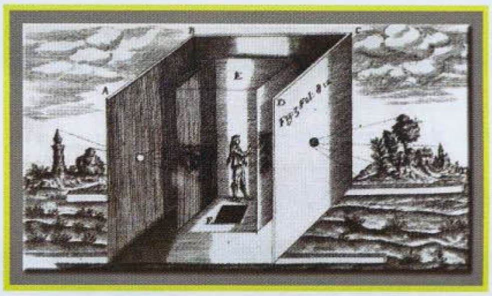


■ **Figure 4.193**
Scene used as an example for ***VRay: Camera*** parameters.

Camera STANDARD

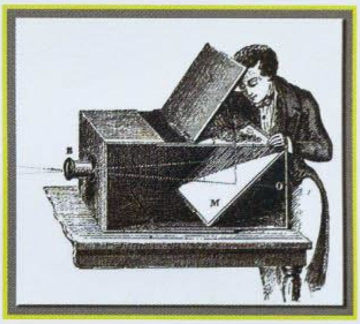
■ **Figure 4.194**
Scene used as an example for ***VRay: Camera*** parameters.

Standard - this is a typical pinhole camera, a stenopeic camera. The first photographic cameras did not have object lens but just a very small hole called stenopeic. Opening and closing it, one could control the flow of light from outside. Behind stenopeic photography lies the concept of "dark room". The possibility of obtaining images via a very small hole that could "capture" images and place it on a flat surface had already been theorized by Aristotle. The term "stenopeic" has Greek origin in fact, and simply means "very narrow". Renaissance painters already used a "camera obscura" (dark room) in order to trace the outlines of landscapes. They would enter the room, which was quite large, and trace the image with chalk, which would be projected on the opposite side of the room compared to where the hole was, upside down and left side inverted with the right side.



■ **Figure 4.195**
One of the first examples of use of the dark room.

The simplest type of dark room was made up of a closed box with a small stenopeic opening on one side which allowed light to enter. This light would project, on the opposite side of the wall the upside down image of what would appear in front of the camera hole. The smaller the opening, the more clear and defined the image would be. The greatest asset of such a simple camera, is that all images were perfectly in focus, regardless of their distance from the camera: in other words its length of focus is infinite. On disadvantage was that not much light could manage to enter the camera so it was only possible to photograph still objects.
Soon, portable dark rooms were invented: the image would be projected onto a mirror inclined by 45 degrees, which would send the image onto a ground glass which was in the upper part of the camera box: basically the principle of reflex. The advantage of the stenopeic hole was that the image did not have to be focused and so depth-of-field was absolute. The disadvantage was that image quality was low: if the hole was large the rays entering would overlap and form a blurry image; if the hole was narrower, then the photograph was more defined but luminosity was much too low.

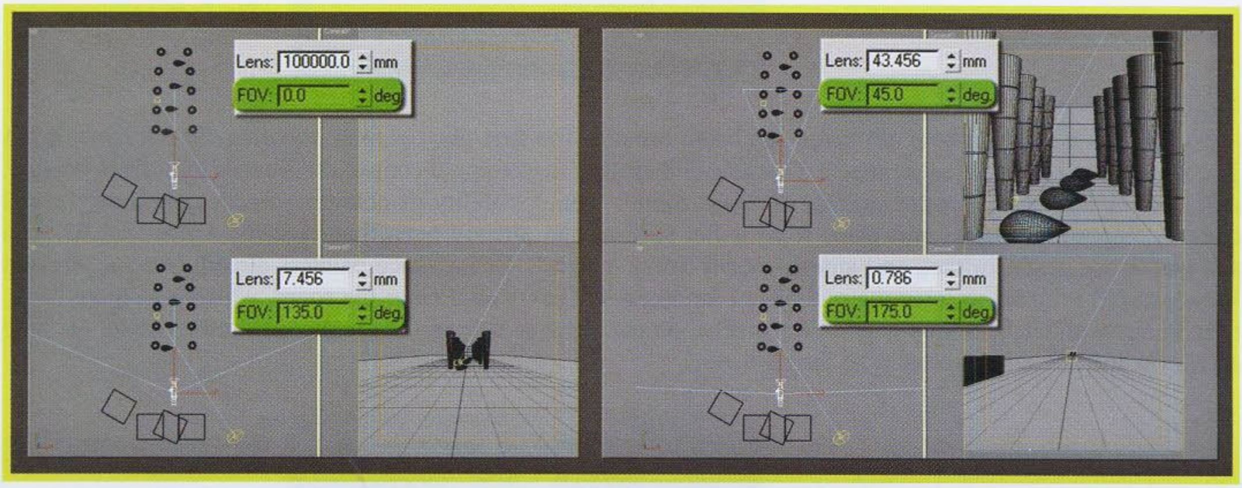


■ **Figure 4.196**
A portable dark room.

For this reason, around 1550, the mathematician Gerolamo Cardano, had the idea of placing a biconvex lens inside the hole. This way the gap could be larger and grant higher luminosity, whereas the lens allowed the focus of rays to be reasonable. From this moment onwards, the history of camera photography begins. The first photograph was taken in 1823 by Nièpce, of a slate of pewter and tar.

■ **Figure 4.197**
Niepce and the first photograph in history.

When using the standard camera, *VRay* renders the scene with the 3ds Max camera, without any type of change. What you see in the viewport is what is going to be rendered. For this type of camera however, one can modify the ***FOV***. By activating it the 3ds Max camera is overwritten *VRay*'s camera. This parameter has been introduced because the ***FOV*** of 3ds Max has a limited range, which goes from 0° to 175°. By using *VRay*'s camera one can employ wider angles.



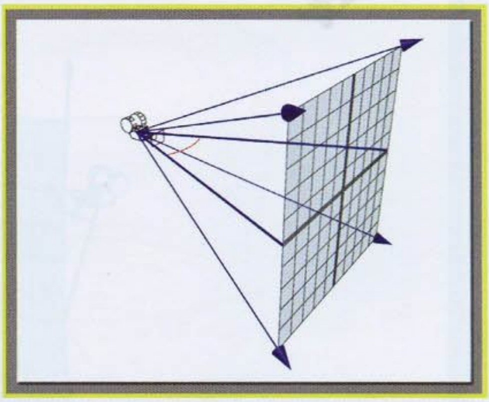
■ **Figure 4.198**
Min and ***Max FOV*** values which can be used in 3ds Max.

Override FOV - this parameter allows one to overwrite the 3ds Max ***FOV*** with *VRay*'s one.

■ **Figure 4.199**
Rendering examples with use of the VRay ***Standard*** camera.

With *VRay* one can use angles wider than 175°. In order to do this one must check the appropriate option , ***Override FOV*** and set the desired value manually. Notice that changing the options in the ***Camera type*** section of *VRay* does not change anything in the 3ds Viewport. The effects of ***VRay Camera type*** can be seen only after rendering.

In the following image, one can observe the way rays are generated, which define the ***Standard Camera***.

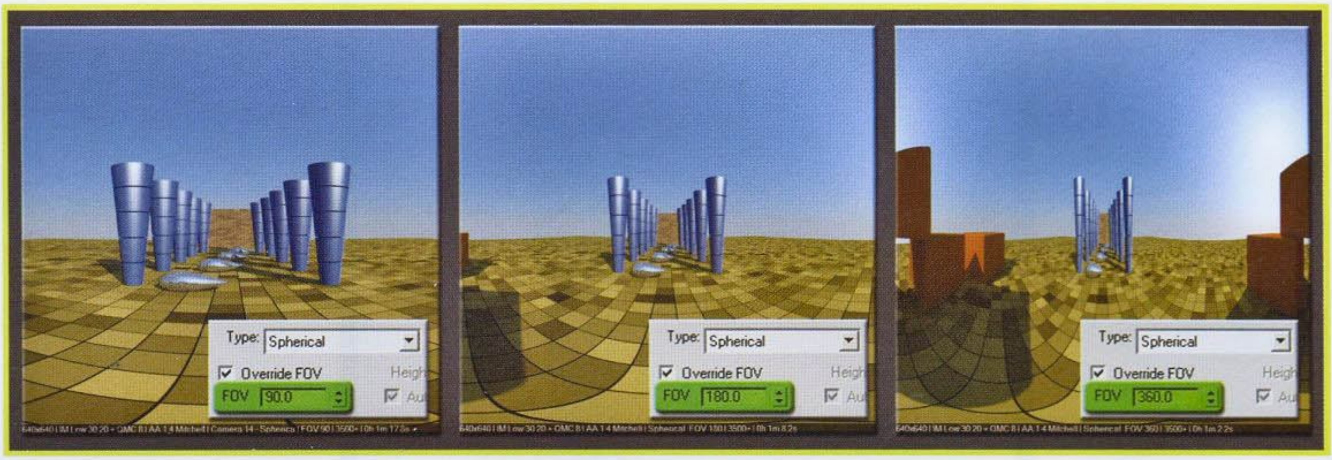


■ **Figure 4.200**
The ***FOV*** is represented by the red angle.

Camera SPHERICAL

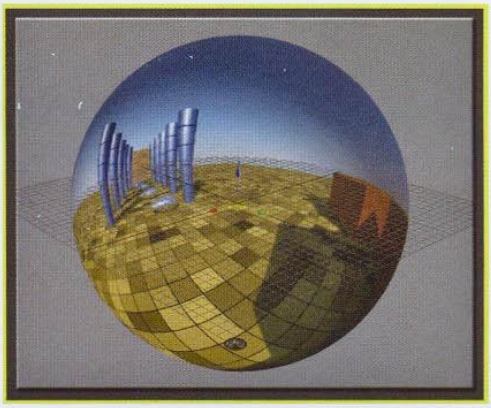
■ **Figure 4.201**
The only available parameter for the spherical camera in VRay is the ***FOV***.

Spherical - in this mode, ***VRay*** renders the scene as if a spherical lens was applied to the 3ds Max virtual camera. As in the previous case, the effects of the new camera are visible only after rendering. By selecting any of the cameras, the viewport in 3ds Max does not undergo any distortions or modifications at all.



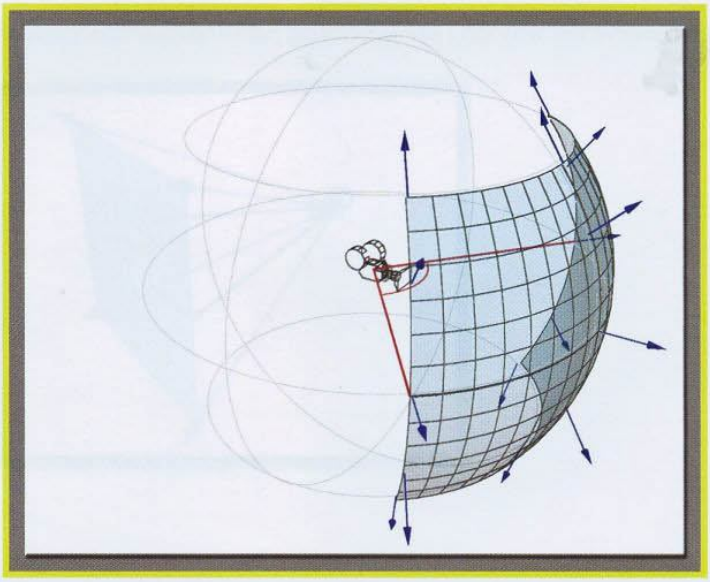
■ **Figure 4.202**
Rendering carried out with spherical camera in VRay.

The render generated with ***FOV*** set to 360° has a particularity: it is an environment. It is a 360° texture, in the sense that it permits one to map a sphere, without generating distortions of any kind. It can be used, for instance, for texturizing the environment of 3ds Max, for mapping a sphere which has the function of a blue sky.



■ **Figure 4.203**
Sphere textured with rendering carried out via the ***Spherical camera*** in VRay.

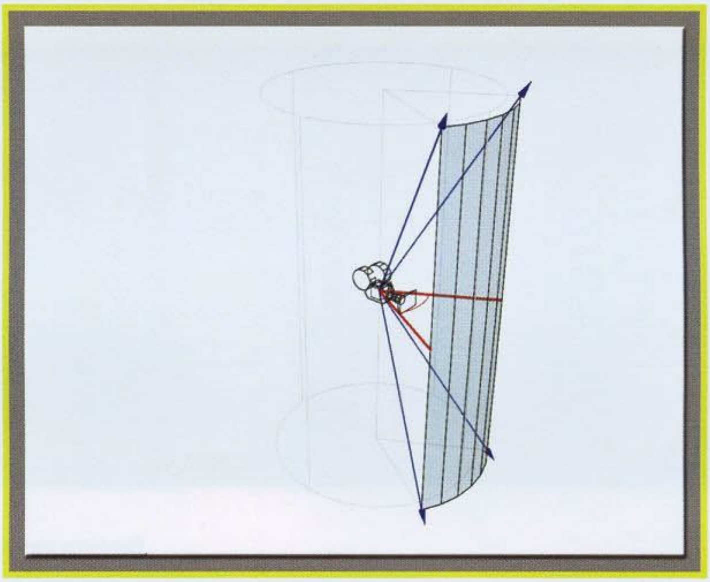
■ **Figure 4.204**
Scheme of the ***Spherical camera***.



Camera Cylindrical (Point)

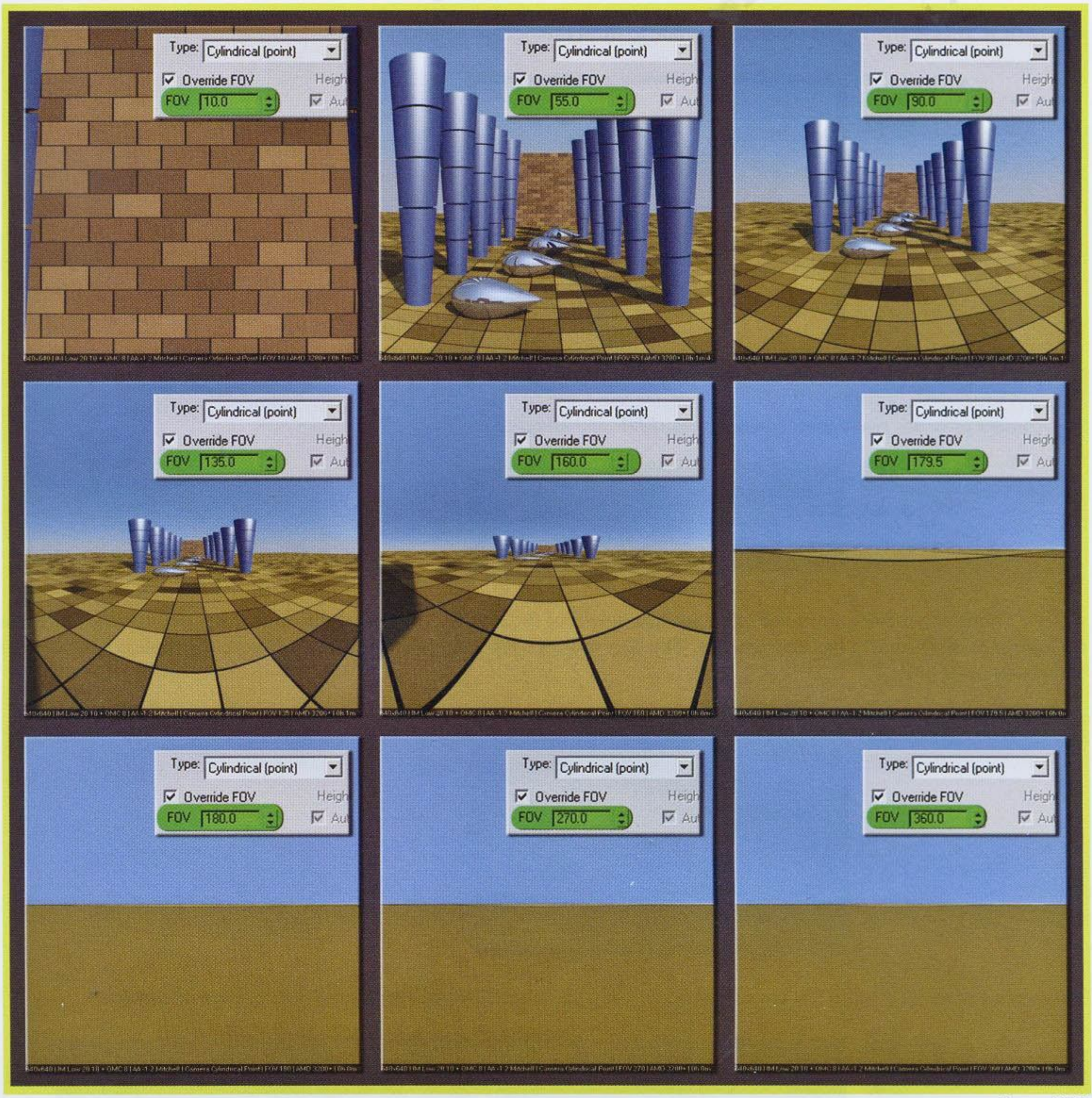
Cylindrical (Point) - with the ***Cylindrical (point)*** camera, rays emitted by the camera have the same origin, as in previous examples. Then they are projected on the vertical surface of a cylinder, instead of on a flat surface which the ***Standard camera*** uses or on a spherical surface which is used by the spherical camera.

■ **Figure 4.205**
Scheme of the ***Cylindrical camera***.



■ **Figure 4.206**
Just like in previous examples, also here the only parameter available is ***FOV***.

Some renders carried out with various ***FOV*** parameter settings with the ***Cylindrical (point)*** camera.

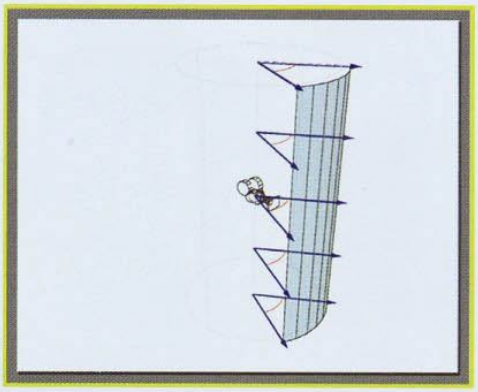


■ **Figure 4.207**
Renders carried out with the ***Cylindrical camera point*** with various ***FOV*** values.

The range of use goes from 0° to 180°. Values above 180° do not change the final render in any way.

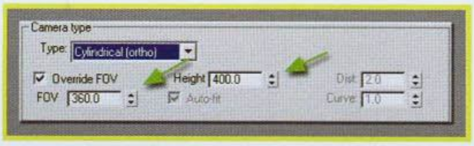
Camera Cylindrical (Ortho)

Cylindrical (Ortho) - this mode allows one to fulfil orthogonal points of view.



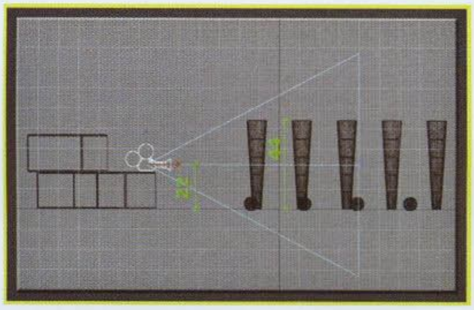
■ **Figure 4.208**
Scheme of the ***Cylindrical Ortho camera***.

This camera provides two variable parameters: ***FOV*** and ***Height***.



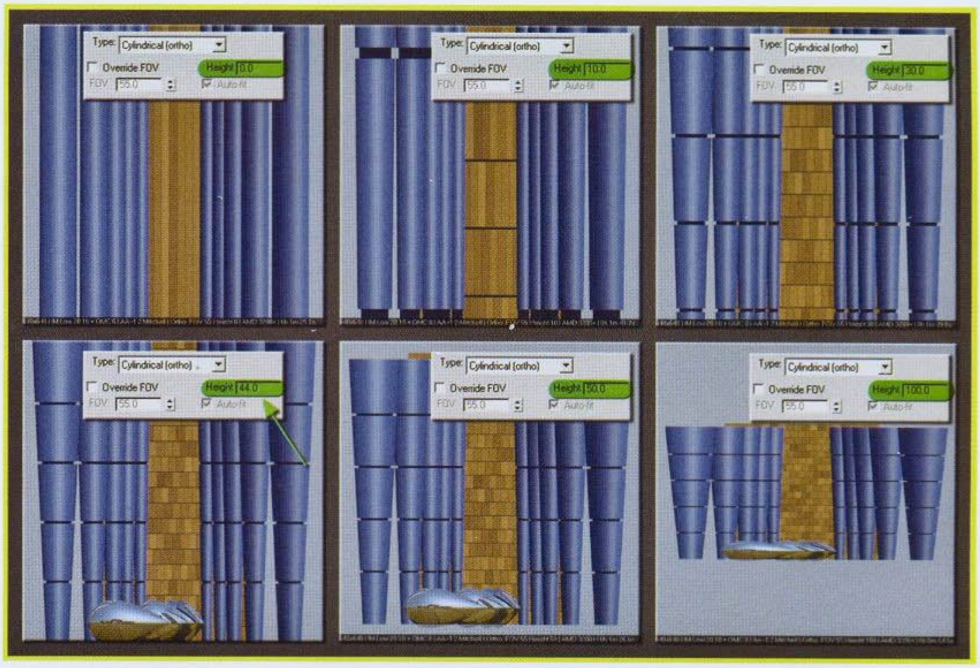
■ **Figure 4.209**
Control panel for the ***Cylindrical Ortho camera***.

Height - this parameter allows one to choose the height of the camera in orthographic mode. It is active only when the ***Cylindical (ortho)*** camera has been chosen. The blue towers have a height of 44 units, whereas the camera has been set to a height of 22 units.



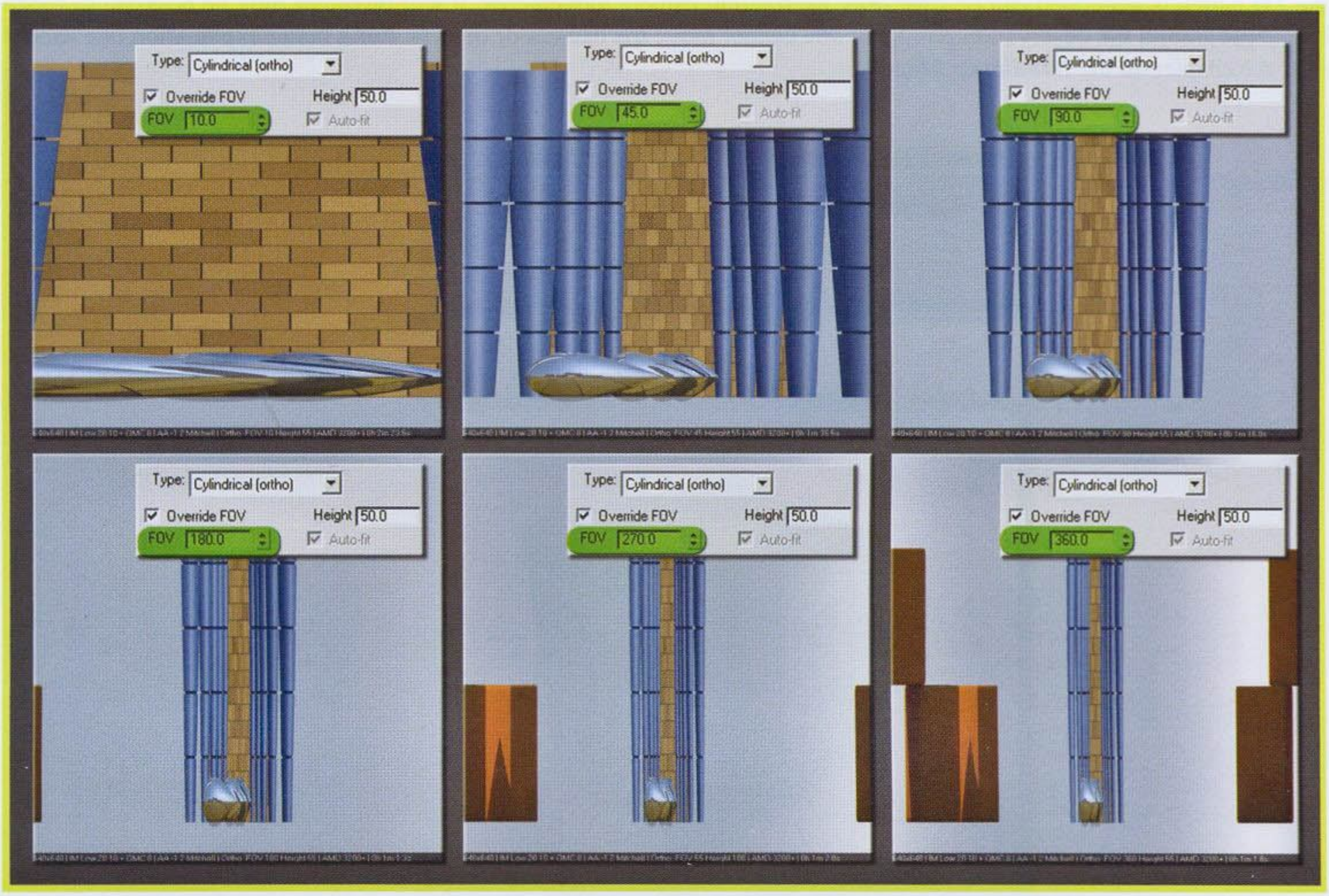
■ **Figure 4.210**
Control panel for the ***Cylindrical ortho camera***.

Some views with FOV set to 55° have been rendered, with only the ***Height*** parameter having been changed.



■ **Figure 4.211**
Some renders with various ***Height*** values.

At this point one can make a few observations: the first is that changing the ***Height*** parameter modifies the render using the centre of the camera itself as the axis of symmetry. This is evident when one observes renders with a ***Height*** value of 44, 50 and 100 units, were "squeezing" the image takes place in the centre of the render. Secondly, with height value at 44, the blue towers coincide perfectly with the superior and inferior limits of the render.
Now we will proceed with modifying the ***FOV*** value, using a ***Height*** value of 50 units.



■ **Figure 4.212**
Rendering carried out with ***Cylindical ortho camera*** with different ***FOV*** values.

If we observe the render calculated with ***FOV*** = 45 carefully one can notice a particularity: although the columns and chrome "teardrops" are the same size, in the render, the objects further away from the camera are thinner. In fact, if you compare a render made with the ***Standard*** camera of *VRay*, but with the 3ds Max Orthographics Projection activated, one can see how the VRay ***Cylindrical Camera (ortho)*** uses an orthographic view for vertical rendering, whereas for a spherical view is used for a horizontal view. This explains this particular distortion effect.

■ **Figure 4.213**
Comparison between the ***Ortho*** option of 3ds Max and the one in VRay.

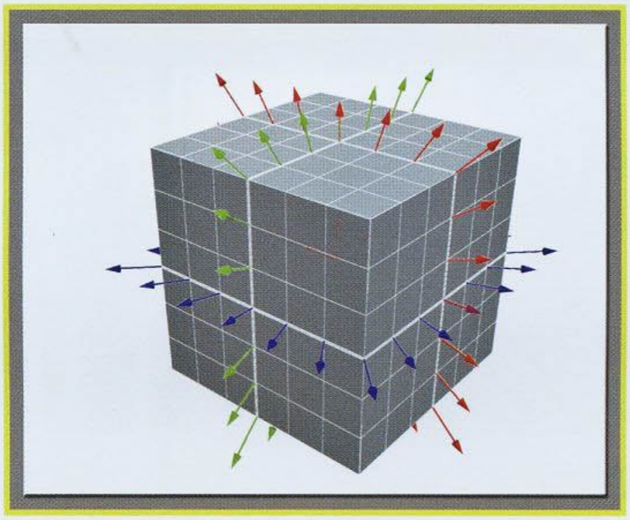
Also, using the Orthographic Projection option, all the *VRay* cameras do not give any results during the rendering phase, and are inactive.

Camera Box

Box - this mode divides the camera virtually into 6 boxes, which render 6 different views, as if the camera were inside a box. This mode is excellent for generating environment maps to be used in Cube mapping mode. It is also useful if one wants to calculate the GI for the whole scene in one go, in order to re-use it for new renders in ***Standard*** mode. But the ***IM*** which is generated can be used only for rotating camera movements. It will therefore only be possible to rotate the ***target*** around the observation point and one cannot move the camera or objects. For a better explanation one should read the chapter on the ***Irradiance Map***.

Figure 4.214
Control panel for the ***Box camera***.

Figure 4.215
Scheme of the ***Box camera***.



The only parameter available in this mode is ***Override FOV***, but this parameter, just like the ***FOV*** in the standard camera in 3ds Max, does not bring any changes.

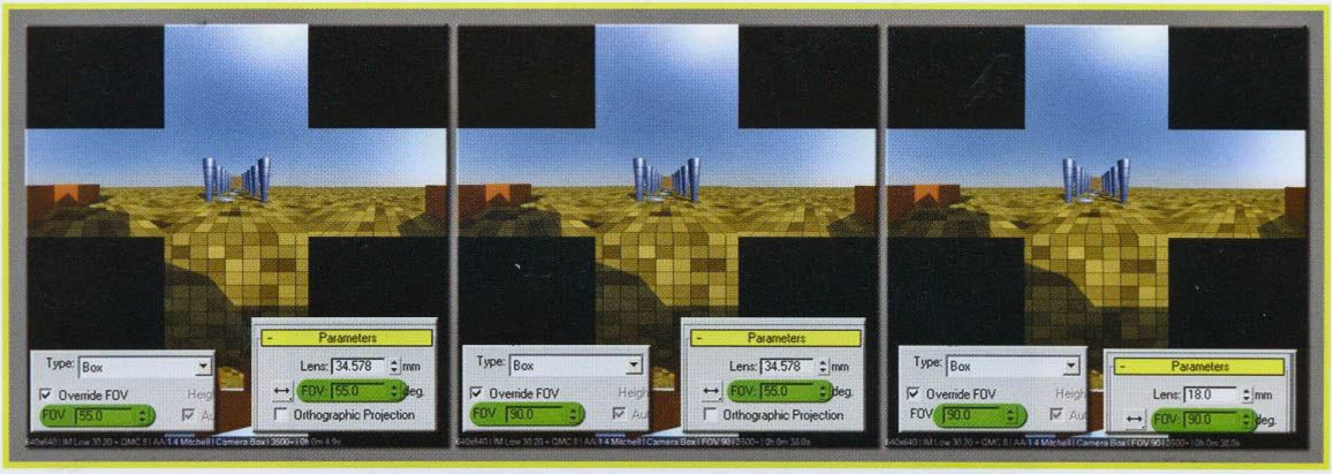
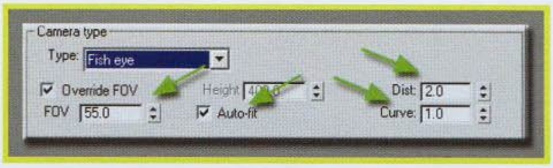


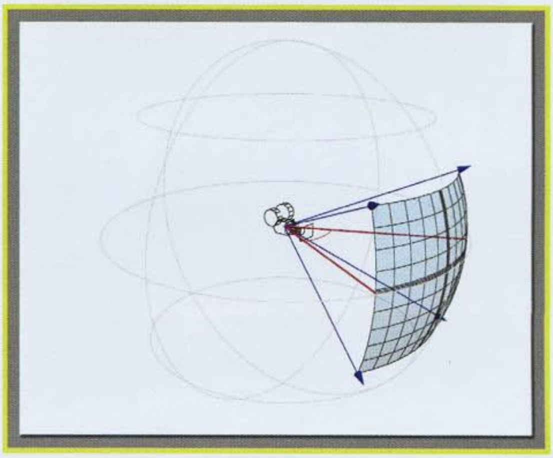
Figure 4.216
Rendering examples made with the VRay ***Box camera***.

Camera Fish eye

This camera is the one with most settings available, four of them.



■ **Figure 4.217**
Control parameters for the ***Fish Eye*** camera.



■ **Figure 4.218**
Graphic scheme of the ***Fish eye*** camera.

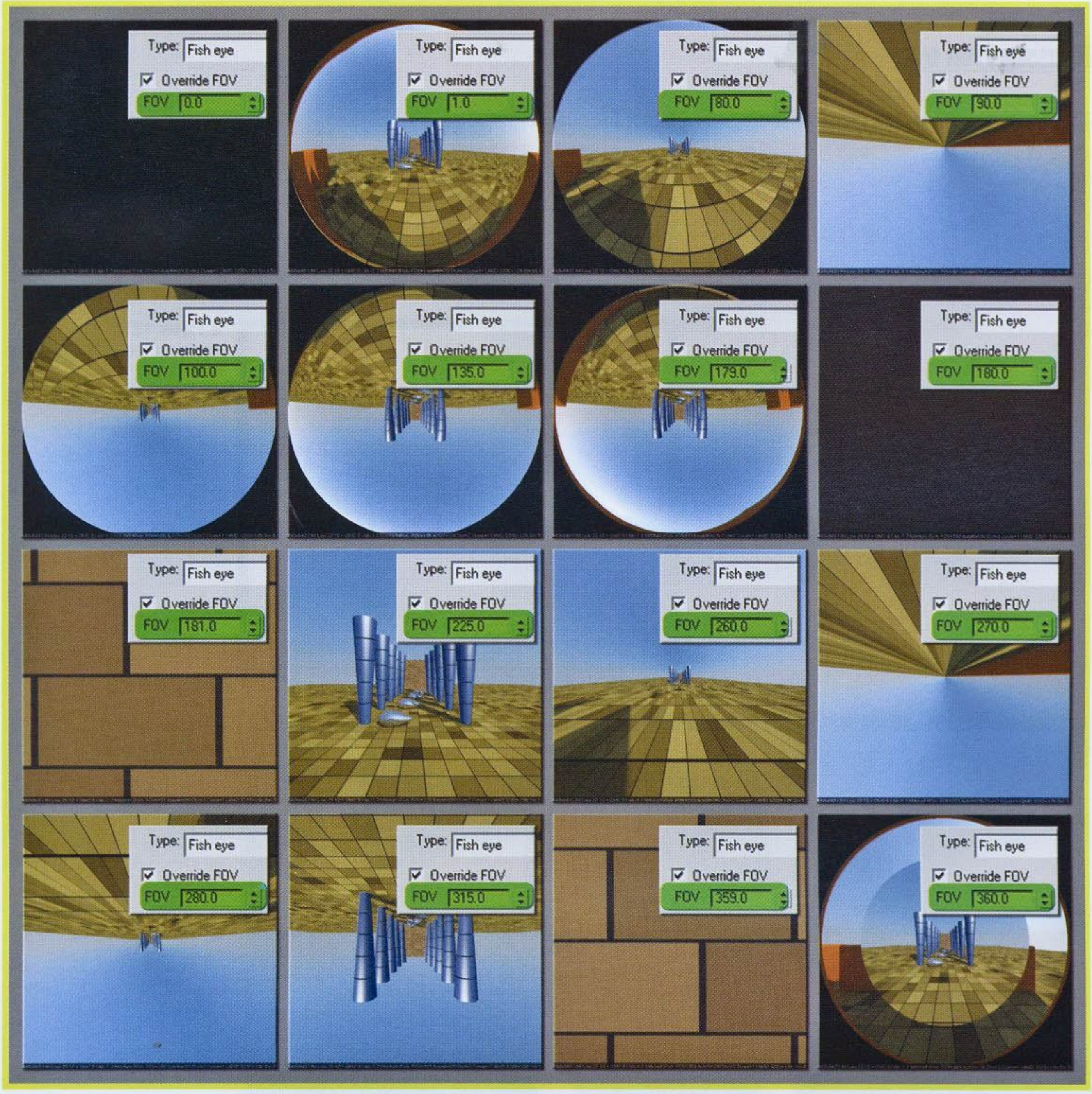
■ **Figure 4.219**
An old ***Fish eye*** lens.

Fish eye - this camera photographs the scene like a normal camera which is however aimed at a sphere which is completely reflective with a unitary ray. It is as if one were photographing a Christmas tree ball decoration. In order to choose which part of the sphere to photograph, one must tweak two parameters: ***Dist*** and ***FOV***.

Auto-fit - this option is available only if the ***Fish eye*** camera has been selected. If it is activated, ***VRay*** automatically calculates the ***Dist*** value so that the render is as large as the output size.

Dist - this parameter is only available for the ***Fish eye*** camera. If we imagine the camera position and the imaginary reflective sphere, the ***Dist*** function regulates the distance between them. This parameter is ignored if ***Auto-fit*** is active.

Curve - this parameter is only available for the ***Fish eye*** camera and controls render deformation. Value 1.0 corresponds to a physically accurate ***Fish eye***. Values under 1 decrease distortion and those above increase it. In other words, this control regulates the angle of reflection for rays that hit the surface of the virtual sphere.



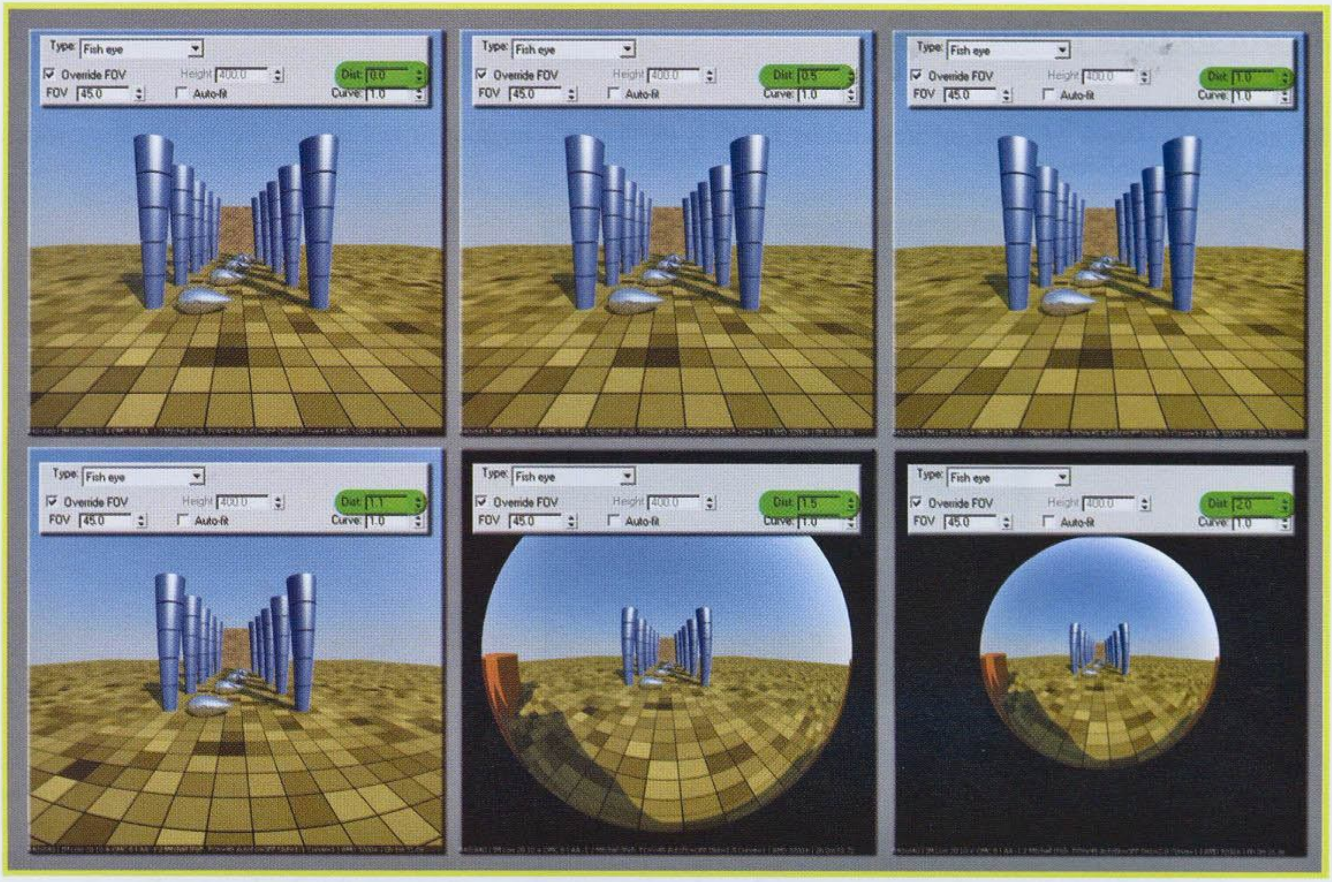
■ **Figure 4.220**
An example of rendering with the ***Fish eye*** camera.

Observing these renders a few considerations can be made:

- The ***Fish eye*** effect evidently only works for angles between 0° and 180°.
- For angles above 180° the camera behaves as if it was a standard camera, except for the fact that above 270° the image is overturned and a strange effect at 360° exactly.

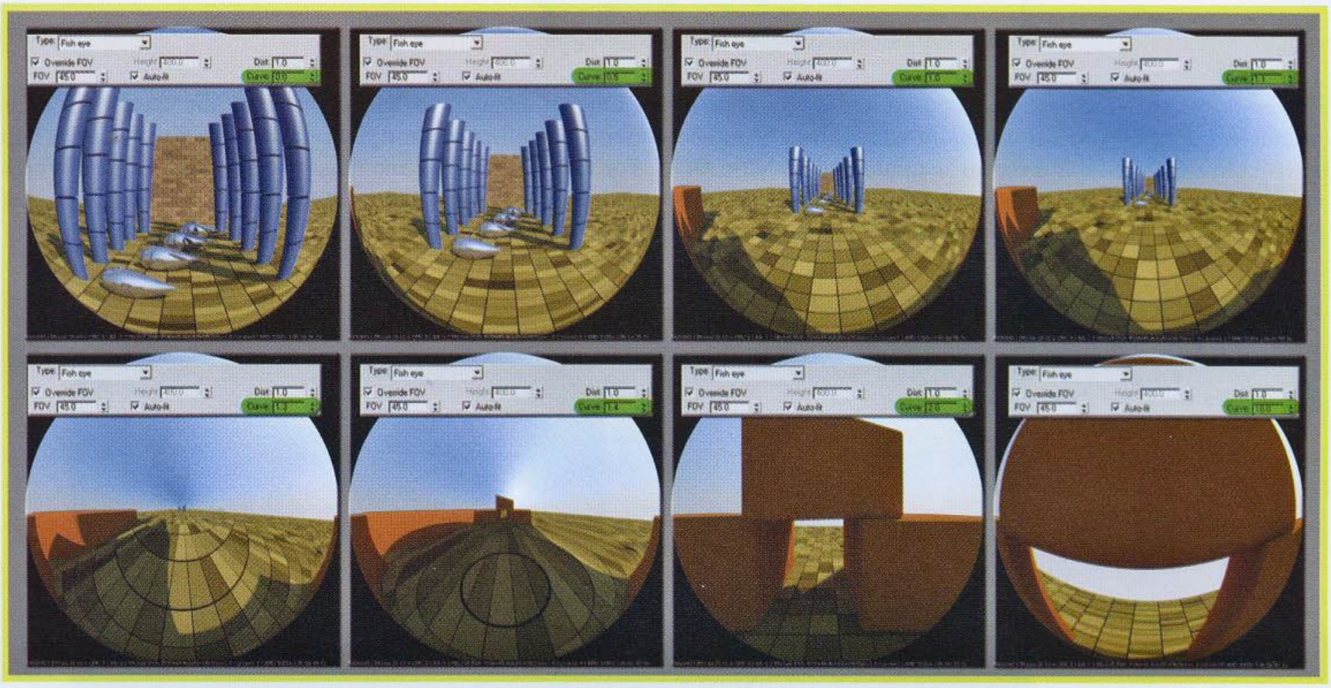
Now some tests will be carried out modifying the ***Dist*** Parameter. In this case one must disable ***Auto-fit***. Changes to the ***Dist*** value affect the render only and not the viewport.

■ **Figure 4.221**
Deactivation of the ***Auto-fit*** parameter.



■ **Figure 4.222**
Rendering carried out with various ***Dist*** values, a parameter active only when using the ***Fish eye***.

It is obvious how values under 1 in the ***Fish eye*** mode do not produce any sort of distortion, because the distance between the sphere and the camera is lower than the diameter of the sphere itself. Values above instead, produce the classic ***Fish eye*** effect. In this case, with values above 1.3, the sphere moves further away and does not completely fill the render.



■ **Figure 4.223**
Rendering carried out with various ***Curve*** values, a parameter active only when using the ***Fish eye***.

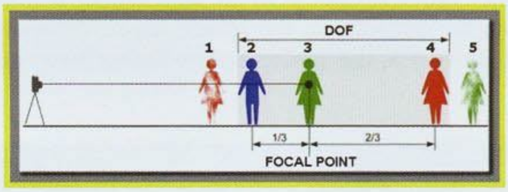
It is obvious how the ***Curve*** parameter influences lens distortion. In some cases the camera is able to frame the back of its target. The correct physically accurate value is however Curve = 1.

Depth of field (DOF)

In photography ***Depth of Field*** (abbreviated as ***DOF***) is the distance in front of and behind the main subject which appears as sharp and focused.

■ **Figure 4.224**
Very low Depth of Field.

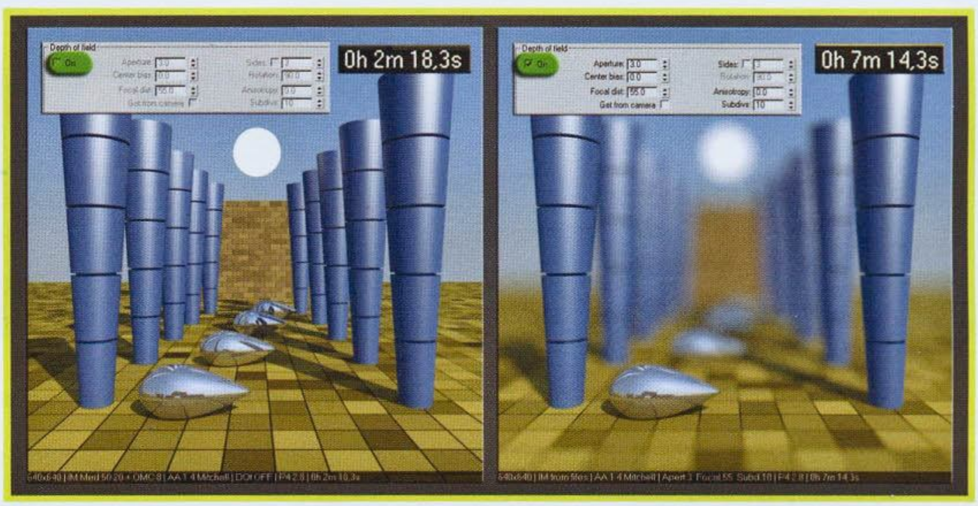
For each setting of the lens there is only one distance where objects appear to be completely focused. The sharpness decreases gradually moving towards and away from the photographer. The depth of field is the distance behind and in front of the subject where blurring is not noticeable or tolerable; the ***DOF*** is defined as greater if this interval is larger and smaller when it is in fact smaller. In the previous example the ***DOF*** is extremely limited. These concepts, from the purely photographical point of view, are explained amply in the section of the book regarding the new ***VRayPhysicalCamera***. For now it is sufficient to understand that by ***DOF*** we mean the zone which is still sharp before and after the plane where the object is focused.



■ **Figure 4.225**
Functioning scheme of the ***DOF***.

VRay allows two methods to be used for ***DOF*** generation. One of them is independent from the camera and its parameters. The user sets it manually with values he or she prefers. The other method, introduced in ***VRay 1.5***, relies on optical physical modelling for an accurate and realistic ***DOF***. For this last method one must use a special camera which ***VRay 1.5*** provides. Here the first method is described, the one which exists since the creation of ***VRay***. The options for ***DOF*** configuration are many and here we analyze them one by one.

On - this parameter activates the ***DOF*** in ***VRay***.



■ **Figure 4.226**
Activation and deactivation of the ***DOF*** effect.

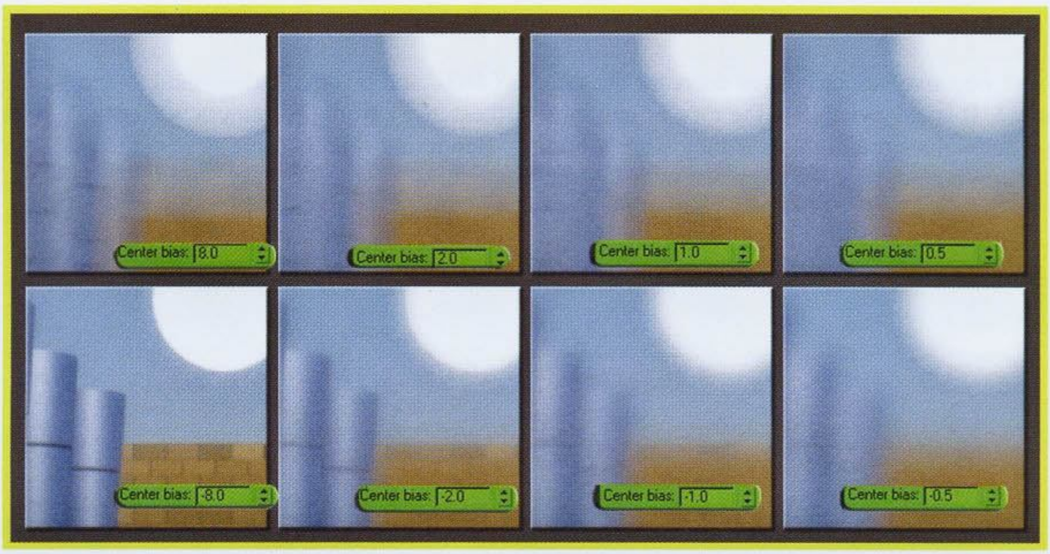
One can see how rendering times are almost threefold and how the scene has become much more realistic. Unfortunately with ***DOF***, rendering times are increased and this is more apparent, the smaller the ***DOF*** is. Soon we will witness how the time for rendering depends on two factors: the opening of the camera and ***Subdivision***.

Aperture - this parameter represents the opening of the virtual camera, measured in ***World*** units. Small values increase the ***DOF***, large values lower it.

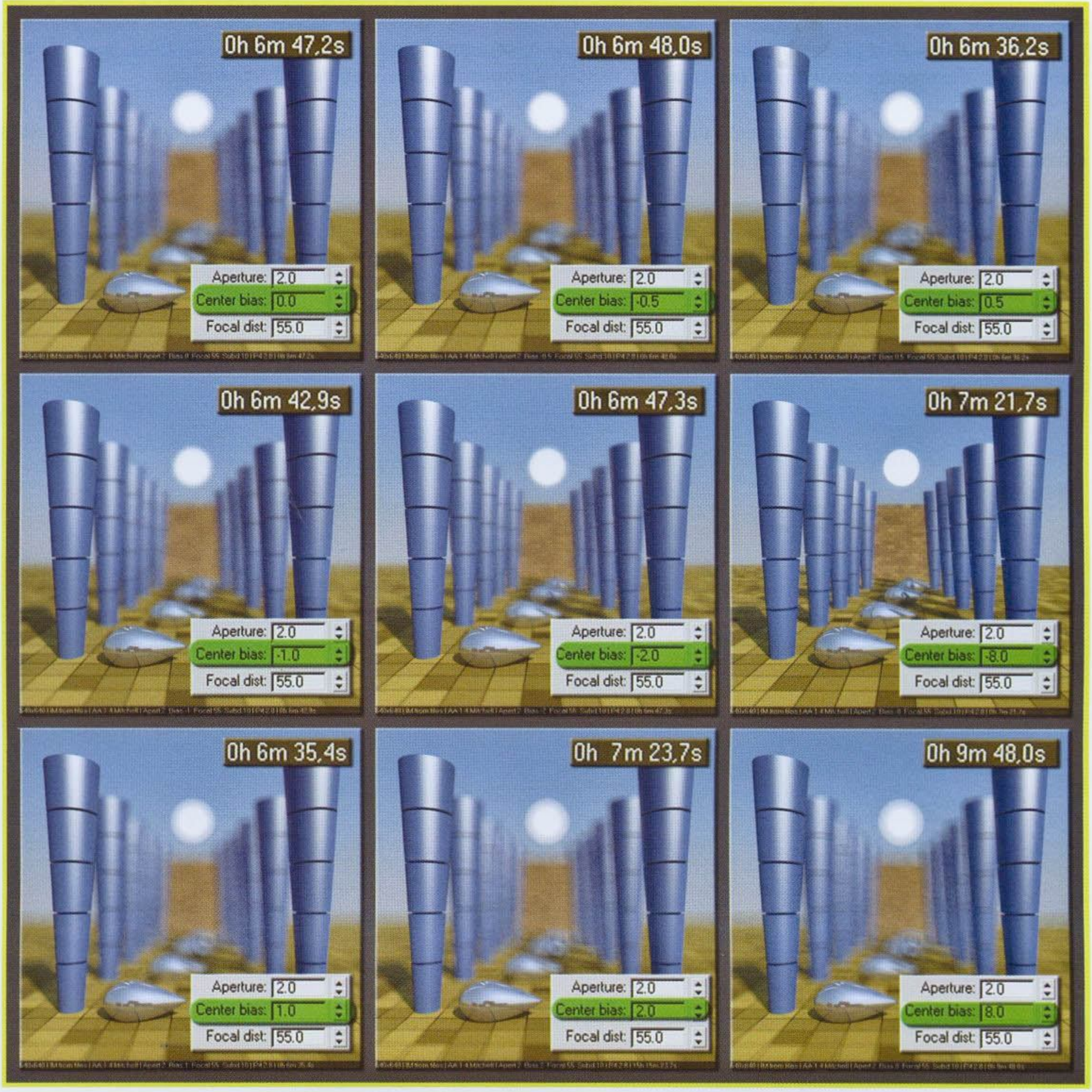
■ **Figure 4.227**
Rendering carried out with different ***Aperture*** values.

If one uses ever higher ***Aperture*** parameter values, the ***DOF*** tends to decrease more and more. Rendering time also increases.

Center bias - this value determines ***DOF*** uniformity. Negative values "block" the ***DOF***.



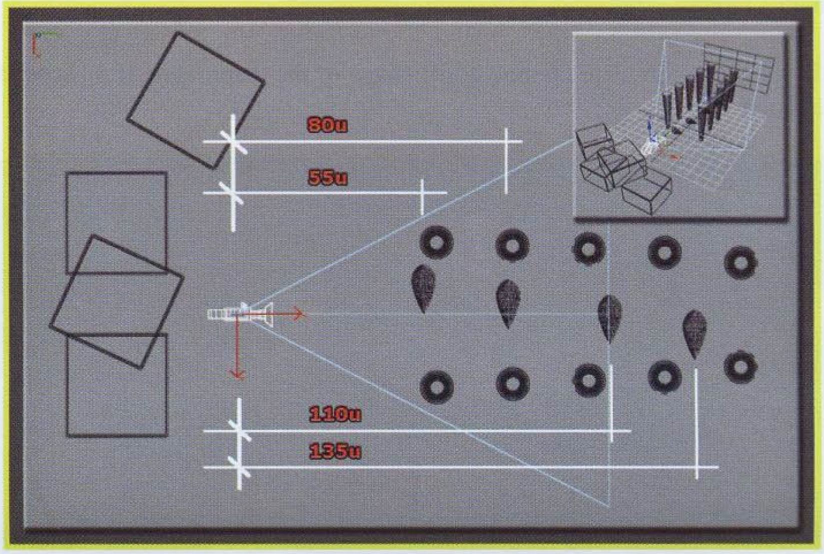
■ **Figure 4.228**
Level of Rendering detail altered with different parameters of ***Center bias***.



■ **Figure 4.229**
Rendered images with different ***Center bias*** settings.

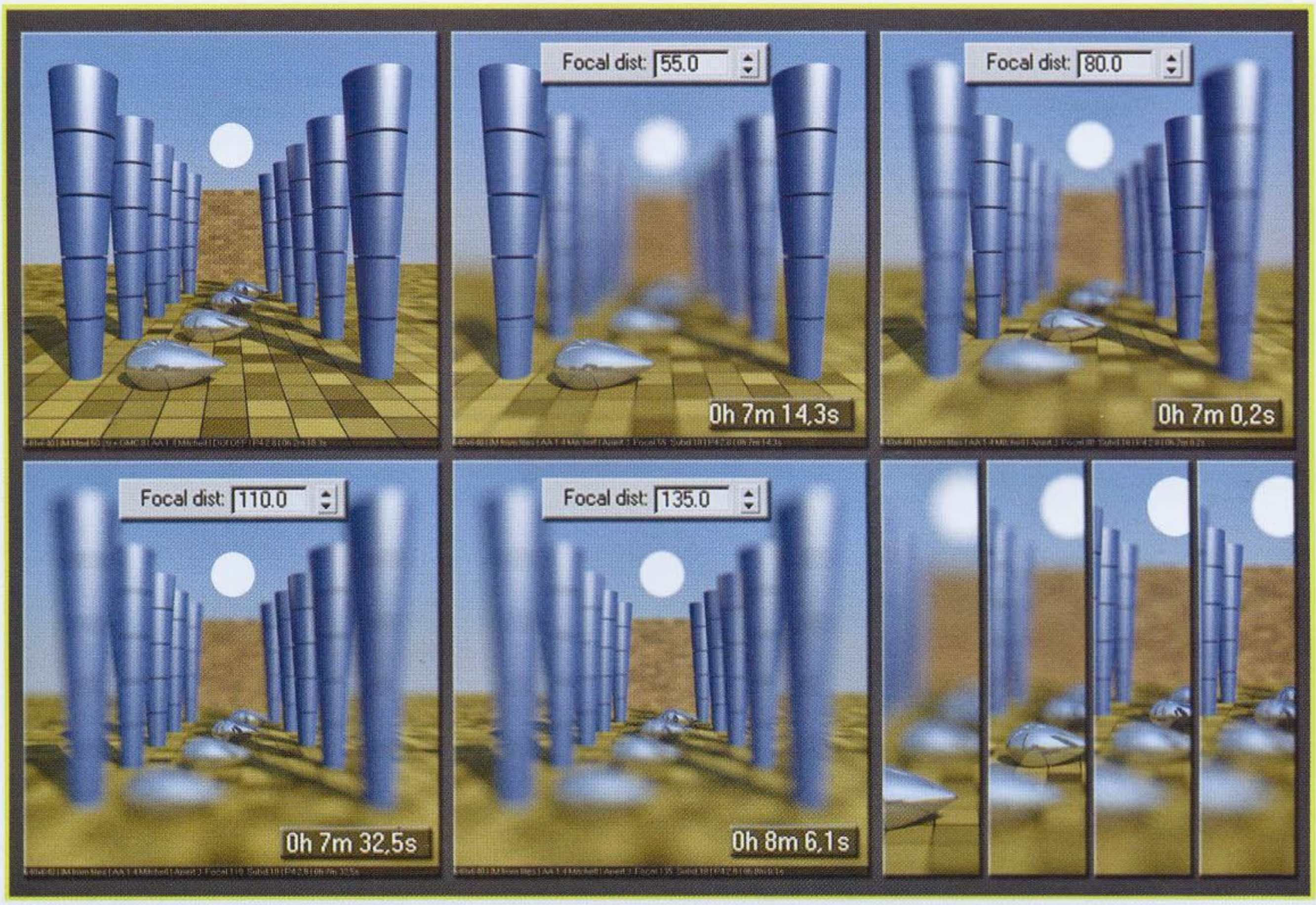
Compared to ***Center bias*** = 0, negative values tend to decrease the effect of ***DOF*** and positive values tend to emphasize it, creating a type of distortion. It is easy visible at the tops of columns furthest away from the camera where the render creates a sort of doubling when values are high, as if they were vibrating. Rendering time is more or less the same for the values considered, except for a peak which takes place with ***Center bias*** = 8.

Focal distance - this defines the distance of the maximum possible focus. Objects which are further or nearer than this distance are not focused. Before passing on to examples, observe the scene and dimensions.



■ **Figure 4.230**
Dimensioned plot of the scene.

In *VRay* there are two methods to determine the exact point of focus. The first consists of analytically specifying the distance one wants the maximum focus to correspond; And this is what the ***Focal Distance*** parameter does. The user must manually input the distance of focus for objects. The other method, instead, consists of using the ***target*** of the camera as a marker for the position for focus.

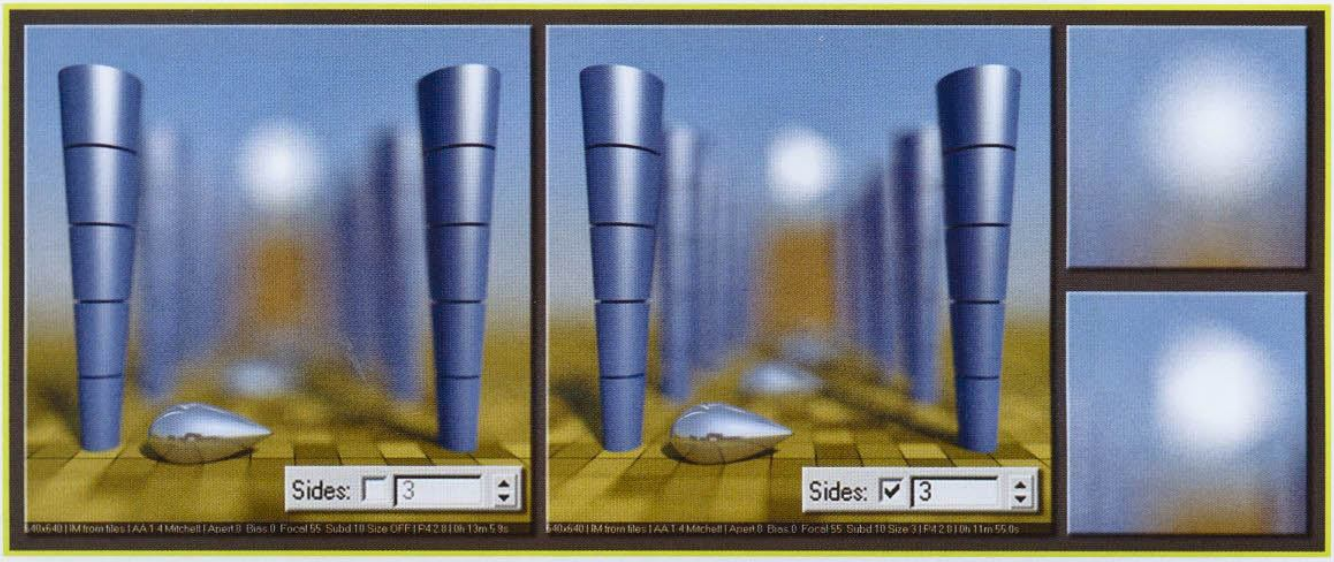


■ **Figure 4.231**
Rendering carried out with different Focal Dist, manually set.

Focal Dist is a precise method for defining the point of focus. Rendering time in this case is around 7 min. for almost all examples. Only in the last case does it last 8 min. This is because the portion of the image where the ***DOF*** is present is smaller and the out-of-focus spots are more intense.

Get from camera - this is the second method for defining the position of top focus. If it is activated this distance is set by the position of the camera's ***target***. It does not work if the render is carried out by a User or perspective view. If this is the case, the values entered into ***Focal Distance*** manually do not influence the ***DOF*** at all, because it is managed by the position of the camera's target.

Sides - this parameter allows one to simulate the polygonal shape of the diaphragm of a photographic camera. If it is deactivated, the shape is circular.



■ **Figure 4.232**
Polygonal effect of the ***DOF*** caused by the activation of the parameter called ***Sides***.

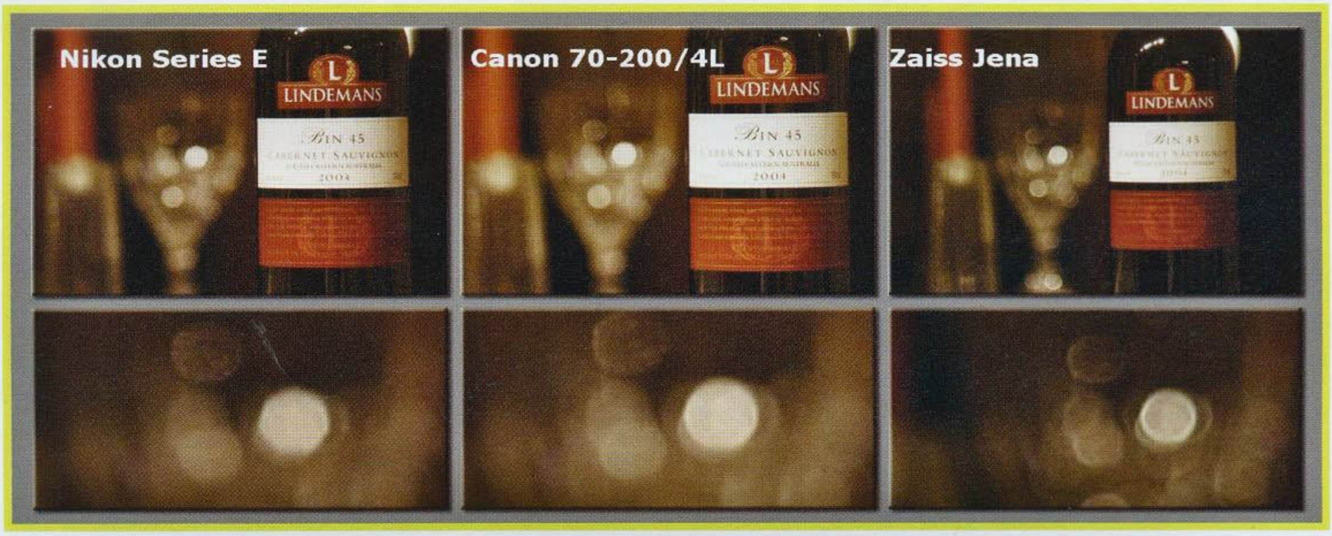
It is easy to observe the effect of activating the ***Sides*** parameter. The luminous sphere, which initially has a completely circular shape, changes its outline when ***Sides*** is activated and tends to take on a triangular shape. This is because the value of ***Sides*** = 3. Values under 3 cannot be entered. In fact no geometrical shape can be made up of two sides. The whole scene undergoes changes and by using ***Sides*** = 3 the ***DOF*** is more defined.

Rotation - this specifies the angle of rotation of the polygonal shape generated by the ***Sides*** parameter.

■ **Figure 4.233**
DOF rotation.

The effect caused by changing the ***Rotation*** parameter is easily recognizable when one analyzes the luminous sphere. The rotation by 45° and 90° makes the triangular shape of the ***DOF*** change inclination.

Anisotropy - this option allows one to extend the ***bokeh*** effect horizontally and vertically. Positive values tend to "iron out" the shape vertically. Negative values do the same horizontally. ***Bokeh*** is approximately the sound of a Japanese word. There is no equivalent word in English and it refers to the quality of an image, in the out-of-focus, blurred areas.



■ **Figure 4.234**
Some ***bokehs*** obtained with different lens setups.

■ **Figure 4.235**
Different images with various ***Anisotropy*** settings.

The effects of changing the ***Anisotropy*** are evident, as is the increase in rendering time. What happens is a sort of stretching of the ***DOF***.

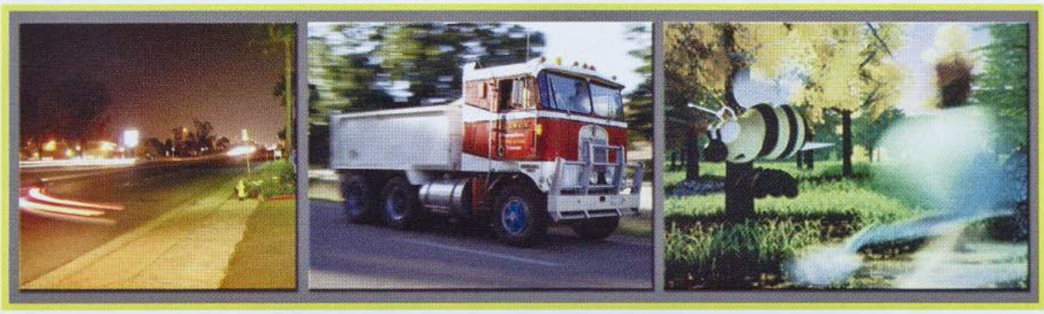
Subdivs - this controls the quality of the ***DOF***. Low values allow a faster calculation, but the ***DOF*** becomes grainy and disturbed. With high ***DOF*** values the quality improved, but rendering time suffers. It is intuitable how ***Subdivs*** depend on the parameters in the ***VRay: rQMC samples***, as in all the effects where the ***Subdivision*** parameter is involved.

■ **Figure 4.236**
Decrease in granularity caused by increase of ***Subdivs***.
- - - DVD - - -

The increase in rendering time is obvious, as is the lessening of noise within images, due to the greater number of subdivisions.

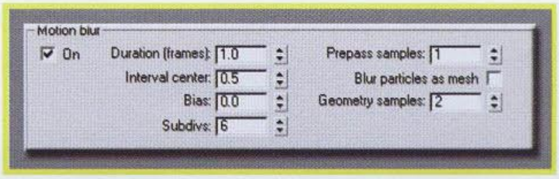
Motion blur (MB)

Motion blur is an optical phenomenon which is created when objects or cameras move fast during an animation. What actually happens is that the image is stretched in the areas of maximum speed.



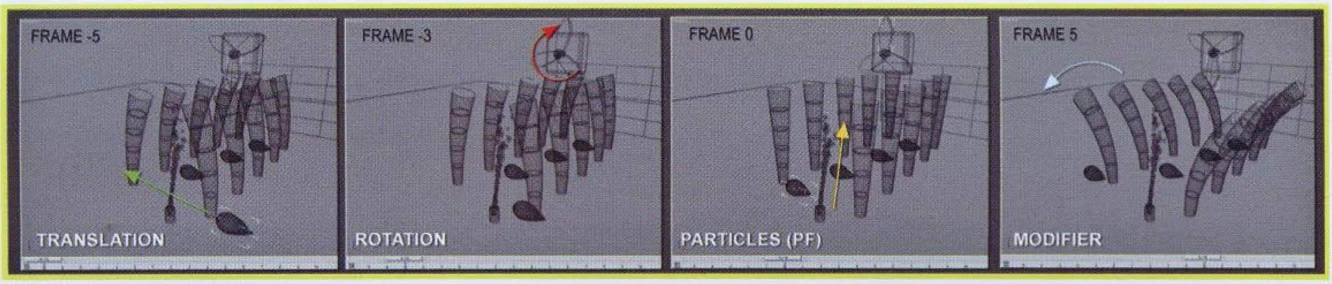
■ **Figure 4.237**
Photographic examples of ***Motion blur***. The car headlights create a long trail caused by the long exposure of the camera. In the second example a truck runs on the road, perfectly focused with a environment in ***Motion blur*** in the background. In the last example we can see a ***Motion blur*** effect in one of the first 3D short films in history.

Motion blur gives animations a lot of realism, but this is also the effect it has in a static image. For this reason, when making animations, or when one wishes to give an object a sense of movement, it is definitely a must to apply ***Motion blur***.

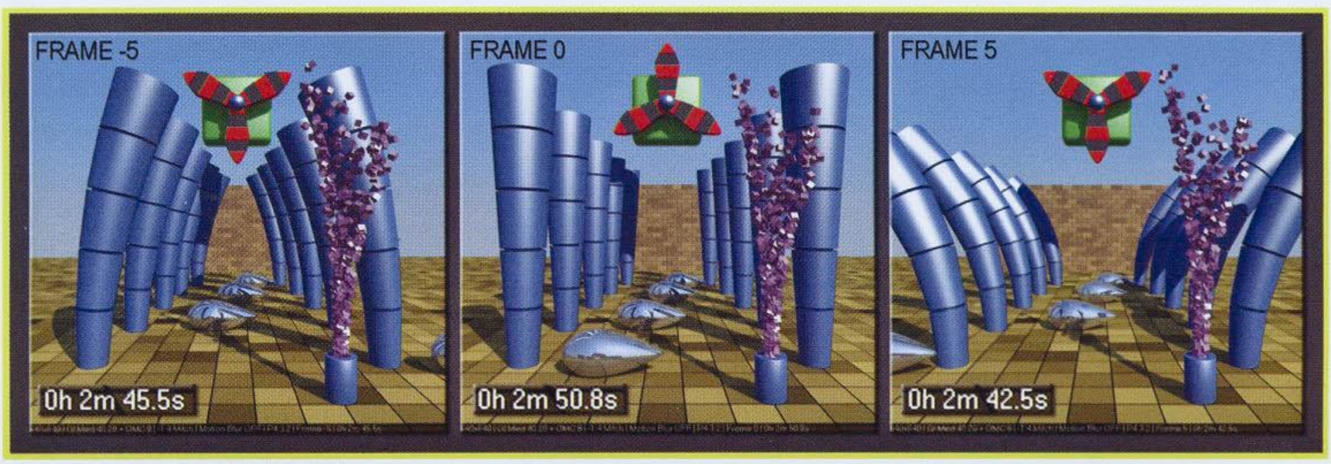


■ **Figure 4.238**
Parameters from the ***Motion blur*** section in VRay.

First of all, we will illustrate the animation being examined. It lasts 10 frames, with an interval which varies from -5 to +5. It is very simple and parts of it move thanks to simple rotations or thanks to the fact of having applied modifiers. Also, a particle system called ***Particle Flow*** has been added which emits moving objects. For a few tests frame number 0 has been used, which is the frame where objects are still. The columns and teardrop-shaped objects are in the same position as in the ***DOF*** test, whereas the fan and the particle system are new objects.



■ **Figure 4.239**
The animation only lasts a few frames, but this is enough to analyze the main parameters which have been made available by VRay.



■ **Figure 4.240**
Rendering of a few frames without ***Motion blur*** activated.

On - this activates ***Motion blur***. All commands and controls are activated or deactivated regarding ***Motion blur***.

■ **Figure 4.241**
Activation of parameters which concern ***Motion blur*** in VRay.

One can see how the use of ***Motion blur*** increases rendering time greatly. Quality and calculation time can be modified according to the user's needs.

Duration - this specifies the opening of the ***shutter*** of the camera, measured in frames.

■ **Figure 4.242**
Variation of the duration of ***Motion blur***.

The effects caused by this parameter are the increased ***Motion blur*** effect and rendering time, although the latter suffers a little from increasing the ***Duration*** parameter.

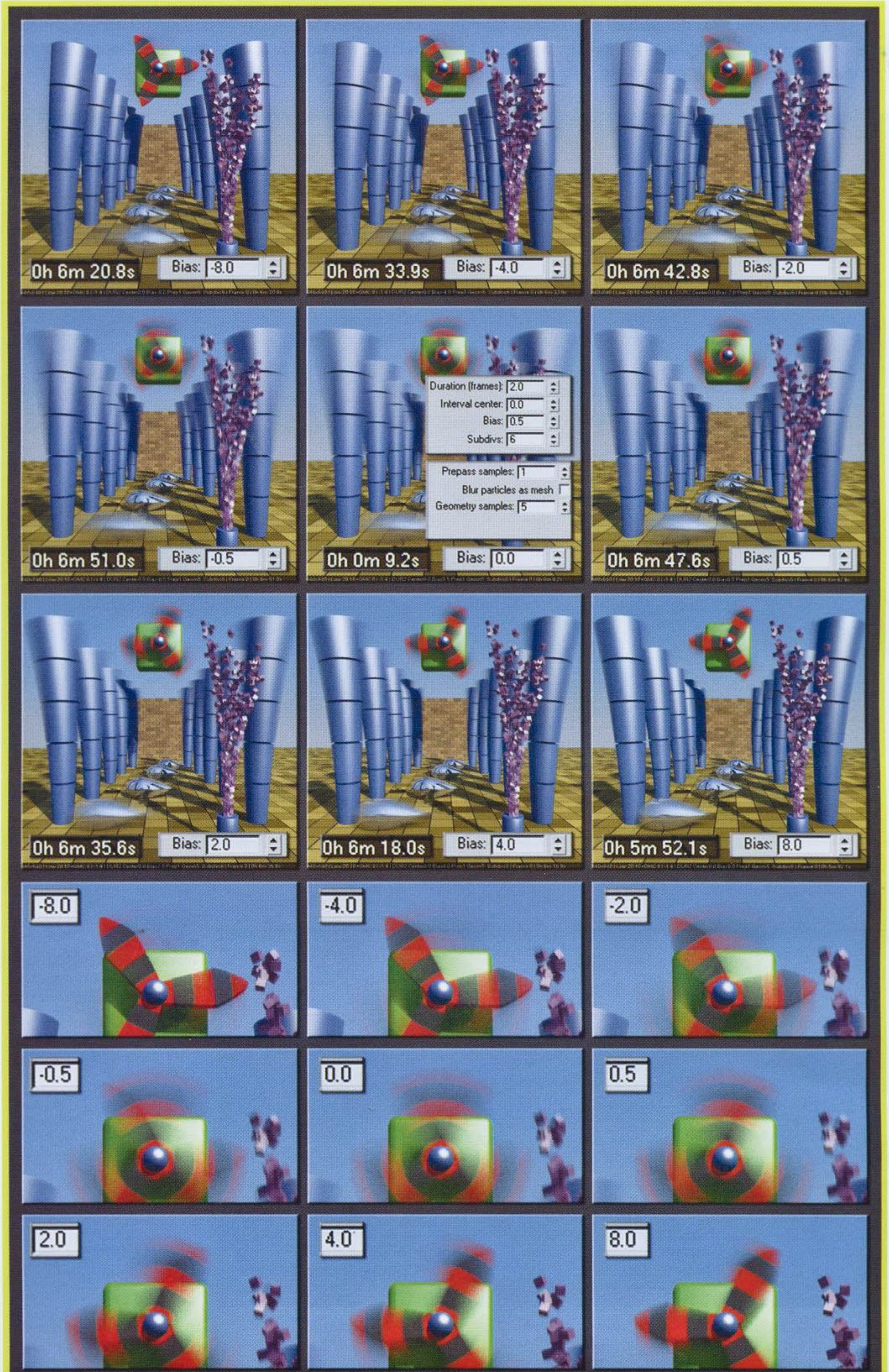
Interval center - this specifies the central position of ***Motion blur*** compared to the 3ds Max frame. Value 0.5 means that the centre of the interval of ***Motion blur*** is between one frame and the next. Value 0.0 means that the centre of the interval of ***Motion blur*** is exactly on the current frame. Default value is 0.5.

■ **Figure 4.243**
Rendering with different values of ***Interval centre***.

This parameter, which has a range from 0 to 1, allows one to move the centre of ***Motion blur*** within the frame of reference, and therefore modify its result. In this case, frame number 0 has been used. With ***Interval center*** = 1 one has the same result that one would get by setting frame number 1 to ***Interval center*** = 0.

■ **Figure 4.244**
With the ***Interval center*** parameter applied on different frames it is possible to obtain the same ***Motion blur*** result.

Bias - this parameter controls the bias of ***Motion blur***. Value 0,0 means that light crosses the camera in a uniform way during the hole interval of ***Motion blur***. With positive values the light is concentrated towards the end of the interval. Negative values concentrate it towards the beginning.



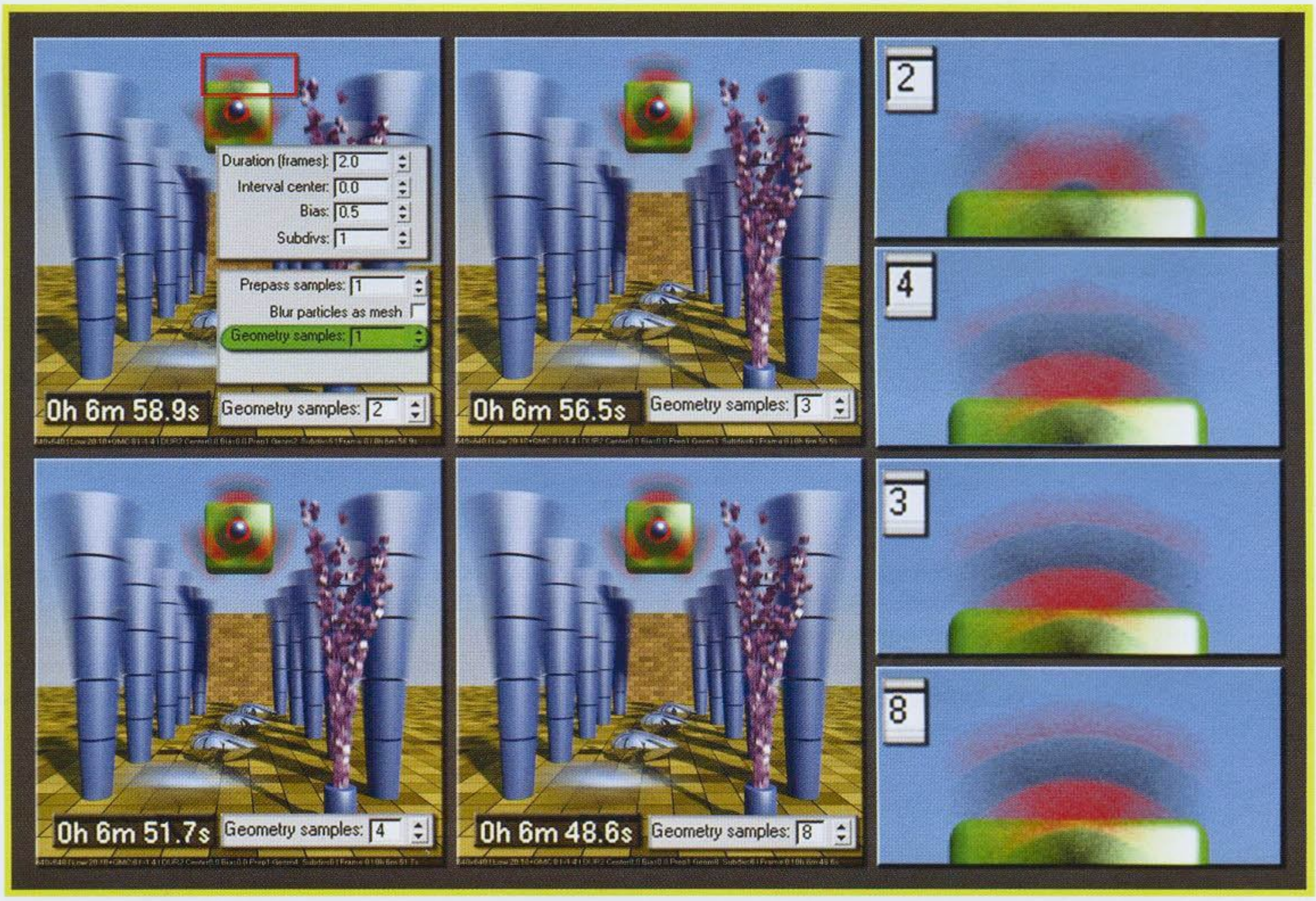
■ **Figure 4.245**
Trail effect caused by changes made to the ***Bias*** settings.

Changing this parameter a trail appears which, in case of negative values, anticipates the movement of the object. It is easy to see how when ***Bias*** = 0, ***Motion blur*** is perfectly symmetrical, whereas with positive values the propeller, behind itself, produces a typical trail of a moving object. If values are too high though, this effect tends to be cancelled. It is therefore necessary to find the correct compromise between ***Bias*** and "trail length". In our case, a value between 2.0 and 4.0 seems to be the best solution.

Prepass samples - this parameter controls the number of samples of the ***Motion blur*** calculated during the calculation of the ***Irradiance Map***.

Blur particles as mesh - this parameter controls the ***Motion blur*** of every single unit in a particle system. When this parameter is activated, particles are considered as single meshes and ***Motion blur*** is applied normally. If the system has its own particular characteristics causing such that the number of particles changes between a frame and the next, this parameter should be deactivated. In this case the calculation of ***Motion blur*** should be calculated during post-production with use of the ***VRayVelocity*** element.

Geometry samples - this parameter controls the number of geometrical segments used for approximating ***Motion blur***. If objects move in a non-linear way one must increase this value, in order to avoid possible flaws. Notice that the more ***Geometry samples*** are used, the more memory is consumed because more copies of the same mesh are necessary to apply ***Motion blur*** to.



■ Figure 4.246
Variation of the ***Geometry samples*** value within VRay's ***Motion blur***.

The effect of this parameter is quite evident, especially on the propeller. As it maintains a rotatory movement, the ***Motion blur*** greatly benefits from an increased number of samples. If it uses only two samples, ***VRay*** only has two instances of the object for calculating ***Motion blur***, at the start and at the end. Interpolation can only be straight. The greater the number of samples, the better the approximation of ***Motion blur*** becomes. If ***Motion blur*** is applied to an object which moves in a straight line, for example the tear-drop object on the floor, the variation of the ***Geometry samples*** parameter does not provide any evident changes.

Subdivs - this controls the quality of noise within ***Motion blur***. Low values permit a faster calculation, but the effect is more coarse and disturbed. With high values, ***Motion blur*** is of a higher quality, but at the cost of rendering time. As all ***Subdivs***, also these depend on the parameters in the ***VRay: rQMC samples***.

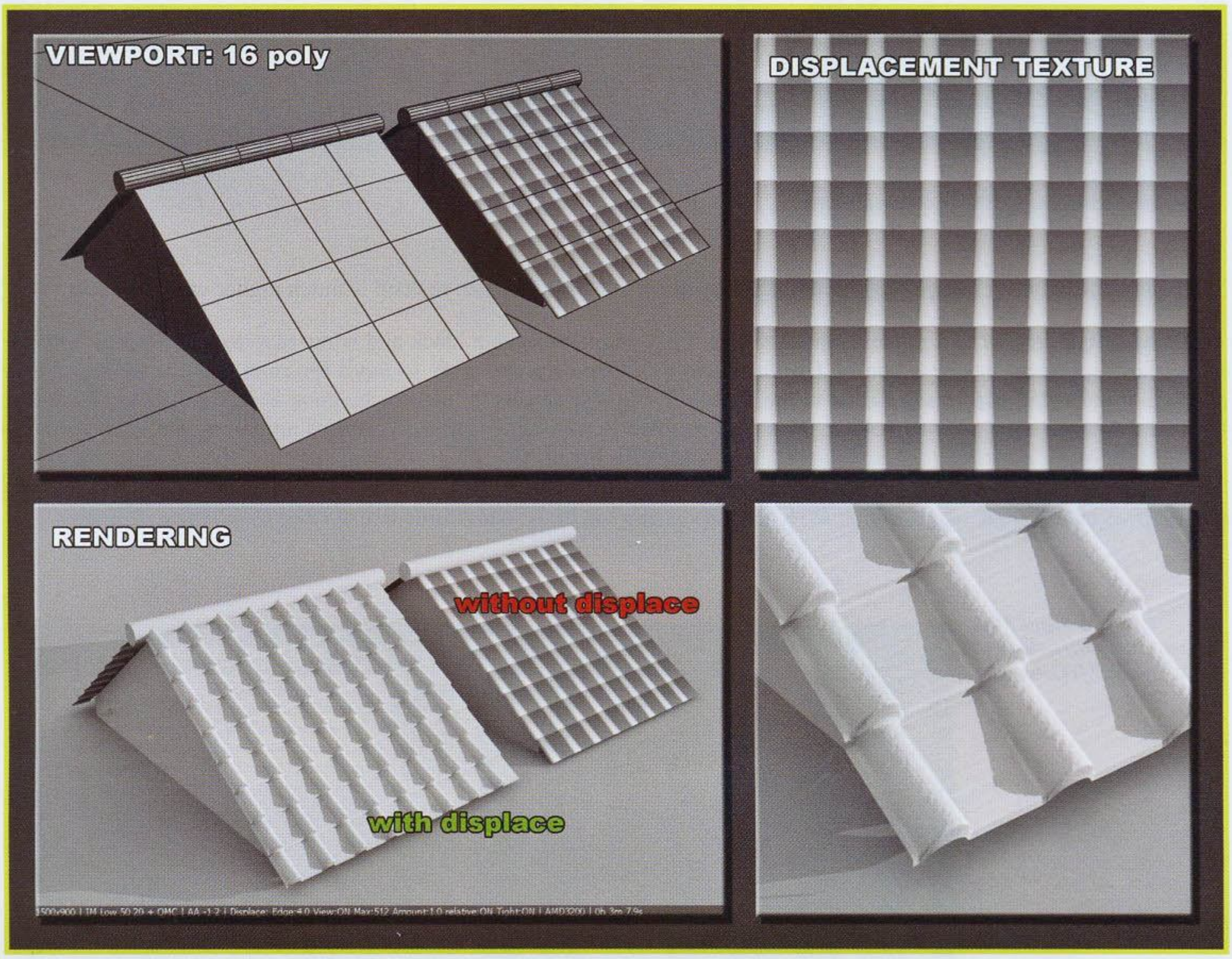
■ **Figure 4.247**
Variation in the number of subdivisions in ***Motion blur***.
- - - DVD - - -

As the number of ***Subdivisions*** increases, the quality improves noticeably, but so does rendering time. The range goes from 4 min. for a low quality ***Motion blur*** to 30 min. for high quality. If ***DOF*** and ***Motion blur*** are being used simultaneously, VRay considers the highest ***Subdivs***.
Often it is useful to fulfil ***Motion blur*** in post-production, rather than directly within 3D software. This allows a greater flexibility and faster renders, but the result obtained in post-production does not always equal the quality one reaches when working in 3D. It all depends on time, the hardware and the quality one wishes to obtain.

VRay: Default displacement

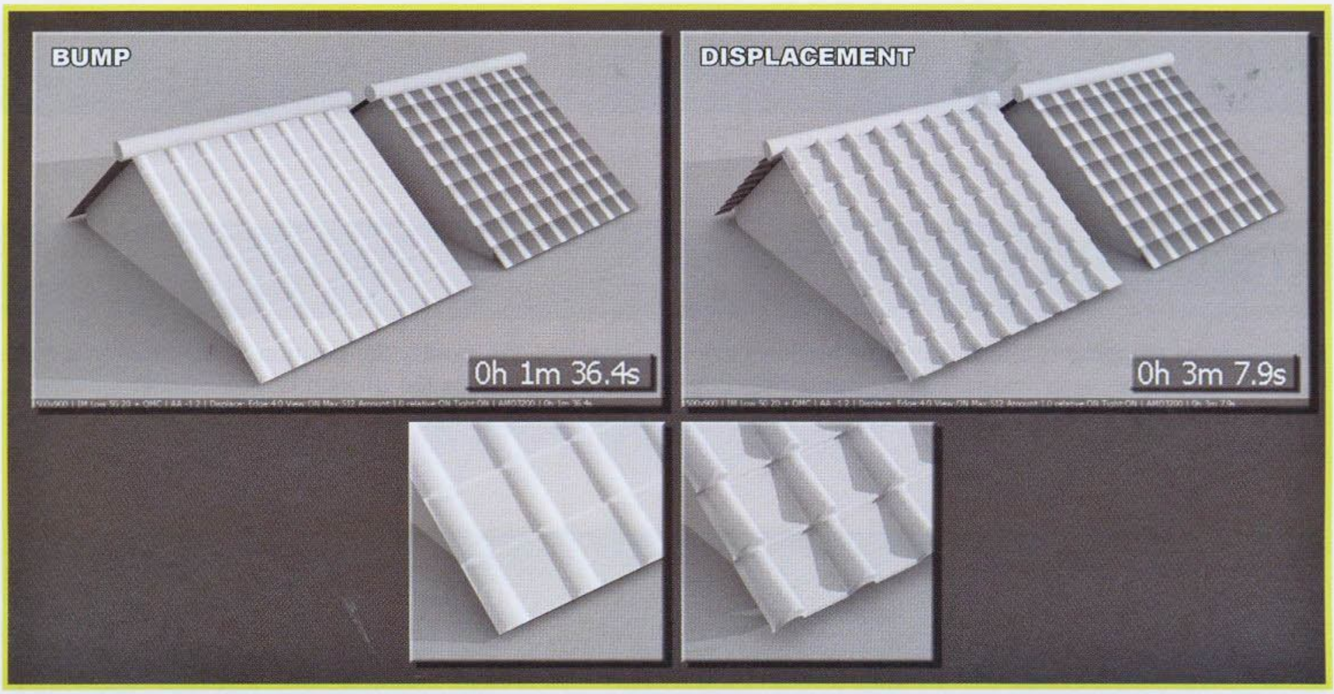
INTRODUCTION

Displacement mapping is a technique which allows one to manipulate and modify the position of the faces in a mesh via a procedural or raster image. Unlike Bump mapping, which only provides the illusion of geometrical movement, ***Displacement mapping*** physically creates new geometrical objects which can generate shadows, reflections or Global Illumination.



■ Figure 4.248
Application of details via a simple black and white map with ***Displacement mapping***.

Although the object only has very few polygons, during the rendering calculation, thanks to ***Displacement mapping***, hundreds of new faces are created which are completely invisible to the user. Their position along the normals of the polygons in question is defined by the value of black and white of the ***Displacement map***. The ***Displacement mapping*** process can be divided into two phases: subdivision into micro-faces (***micro-polygons***) and their shifting. Unfortunately this subdivision only takes place during rendering and is not visible in the viewport. The thicker the subdivision is, the more memory is consumed and thus the more rendering time increases due to the increased geometrical complexity. From the previous image one can observe how the geometrical object created by the ***Displacement mapping*** even influences direct shadows. This confirms the fact that the ***Displacement*** feature actually creates new geometries.

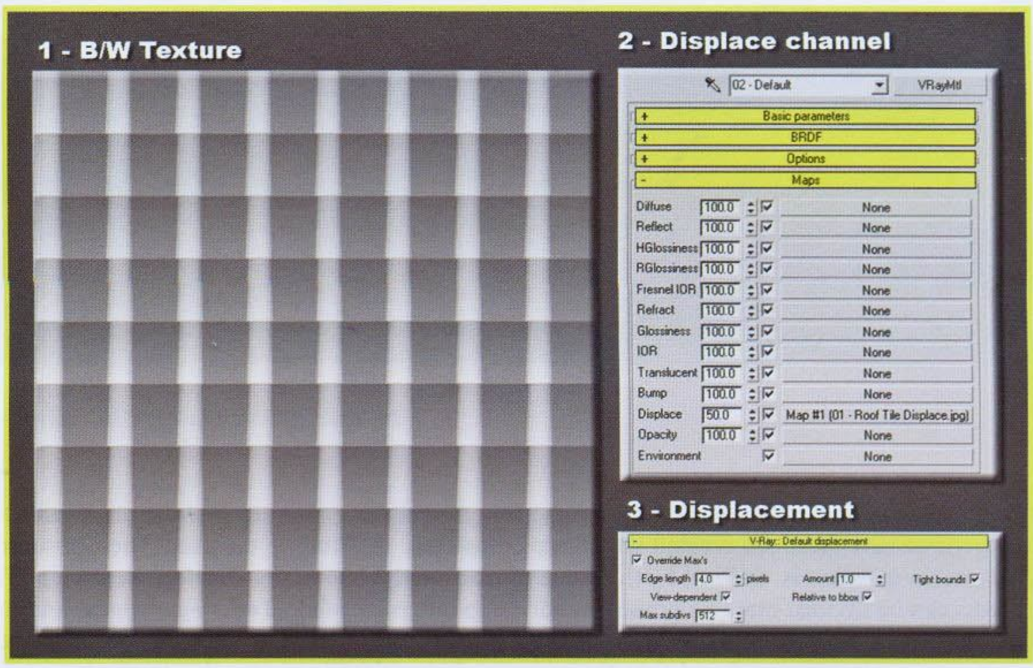


■ **Figure 4.249**
Comparison between ***bump*** and ***displacement*** effects.

The difference between bump mapping and ***Displacement*** is clearly visible: realism is much higher with the latter.

Applying ***Displacement*** to a model is quite simple. First one must have a black and white map to start from, where the white part represents the elevation of ***Displacement*** (1). One then proceeds to input the map into the ***Displacement*** channel of the ***VRay*** shader associated to the model (2). Lastly, all that remains to be done is activate ***Displacement*** in the ***VRay:Default displacement*** rollout. This method must be used when ***Displacement*** is not taking place because of a modifier, in other words as long as there is no ***VRayDisplacementMod*** modifier applied, but only via shaders. This is why it is called ***Default displacement***; because it is a fast system for creating this effect. If one wishes to have a greater control over every single object which ***Displacement*** is applied to, it is advisable to use ***VRayDisplacementMod***.

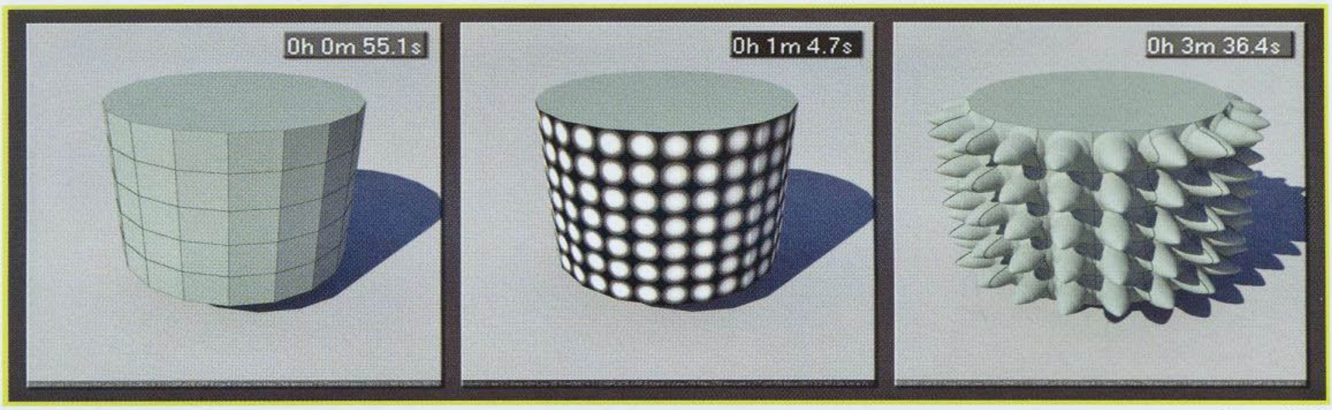
■ **Figure 4.250**
In only 3 steps one can attain an excellent ***Displacement***.



PARAMETERS

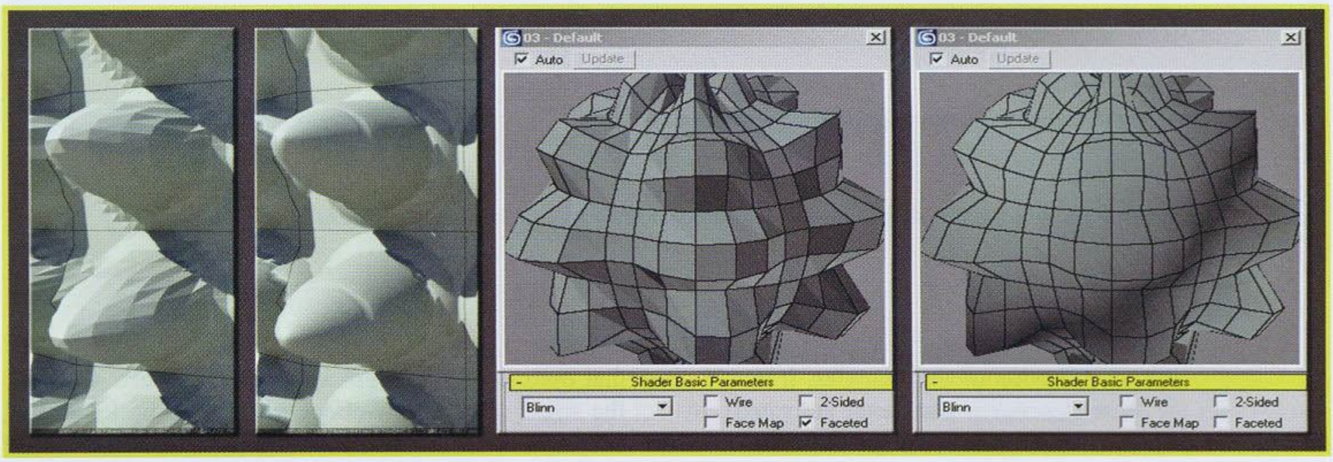
Override Max's - this parameter, when active (by default), causes *VRay* to render objects to which ***Displacement*** is applied using the proprietary system of micro triangulation. If it is disabled and ***Displacement*** is assigned by a Standard shader, the classic ***Displacement*** system of 3ds Max is used. By using a ***VRayMtl*** shader and deactivating the ***Override Max's*** parameter, no ***Displacement*** is processed. In most cases this parameter should be left active, as it does not happen often that one needs to use the 3ds Max Displacement in *VRay*.

Edge length - this determines the quality of ***Displacement***, specifying the maximum length of the side of ***Displaced*** sub-triangles. Thanks to ***Displacement*** each triangle which makes up a 3D model is divided into microtriangles. Although the subdivision process is invisible to the user, one can in any case decide the number by fixing the length of their sides with the ***Edge Length*** parameter. The lower the value, the more triangles are generated. Therefore ***Displacement*** has higher quality but the calculation requires more time and consumes more memory. Its behaviour also depends on whether the ***view-dependent*** parameter is enabled.
The scene being examined is very simple. It is a cylinder made of 18 polygons on the circumference and 5 height-wise, to which a ***Displacement*** map is applied formed by circular gradients in B/W, repeated many times.



■ **Figure 4.251**
Scene used for various illustrative tests for explaining ***Displacement***.

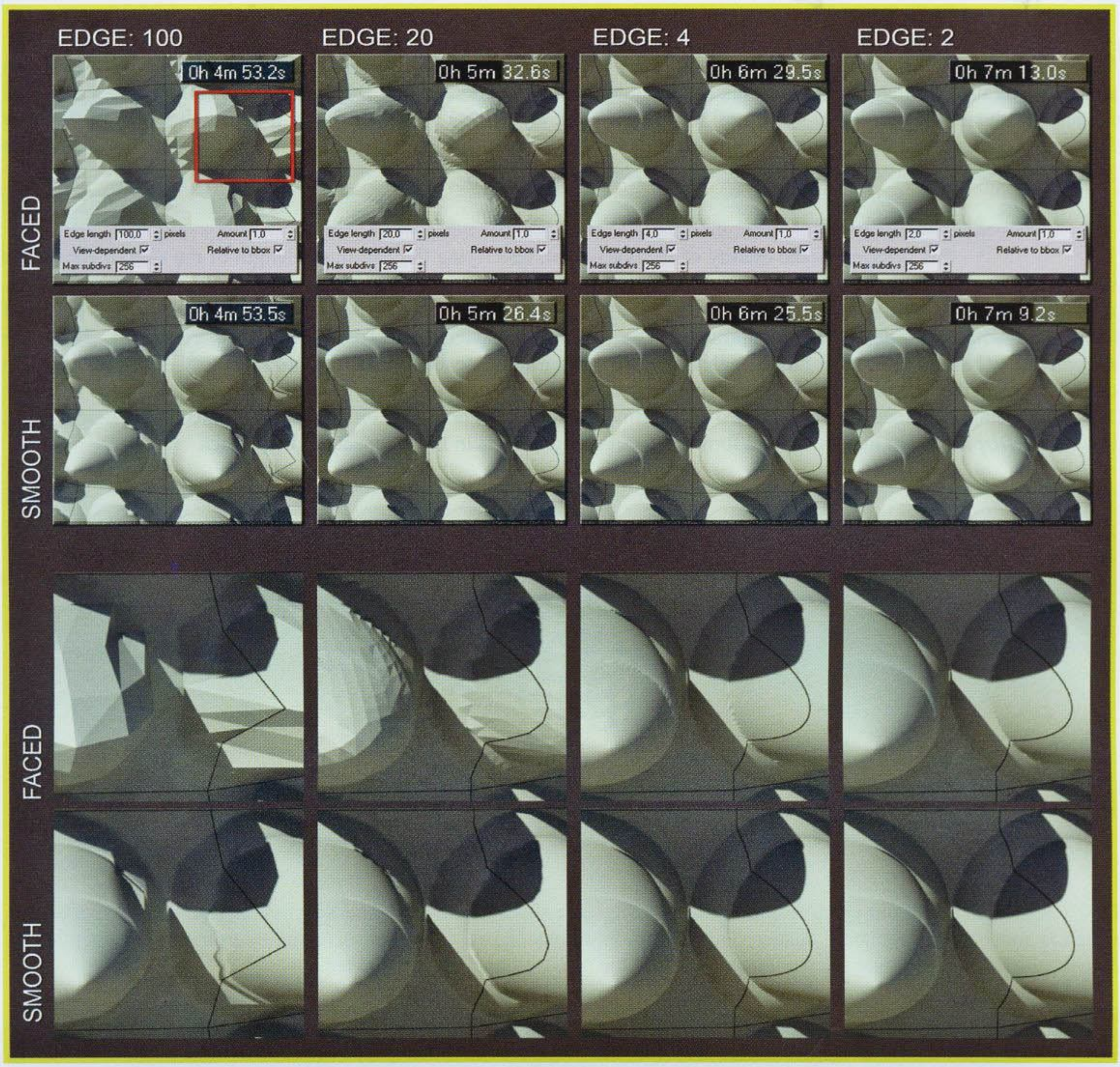
Firstly tests are carried out using two variations of the same shader, activating and deactivating the Faceted parameter, which is found in the Standard shader. This operation allows one to clearly observe the microtriangles generated by ***Displacement***. By activating the parameter, one sees the result without microtriangles.



■ **Figure 4.252**
Enlargement of the zone which is being tested with the same ***Edge length*** value. In the Shader Editor slot one can observe the effects of the ***Faceted*** option.

Already in these first examples we can observe how the resolution of the subdivision in microtriangles does not correspond to the smoothed out version. In order to compensate for the low resolution of ***Displacement***, *VRay* automatically generates a ***Normal map*** based on the information obtained from ***Displacement***. This is applied in the rendering phase to improve quality further.

It is clear that the user must define the correct number of microtriangles, so that there are enough, together with the generated ***Normal map***, for a good quality render.



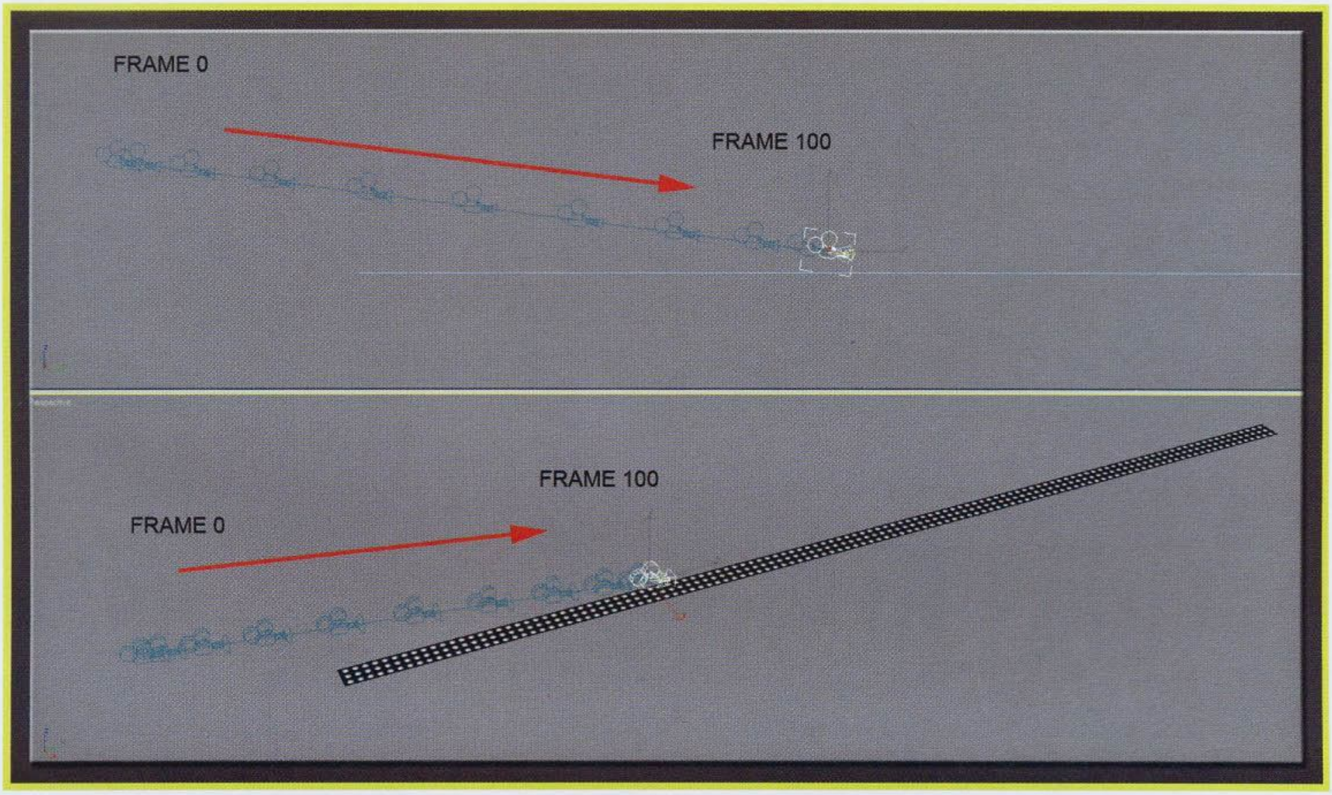
■ **Figure 4.253**
Comparison between different ***Edge length*** values using faceted and smoothed shaders.

This test shows some factors which contribute to the goal of acquiring a good ***Displacement***.

- Triangle size.
- Rendering time.

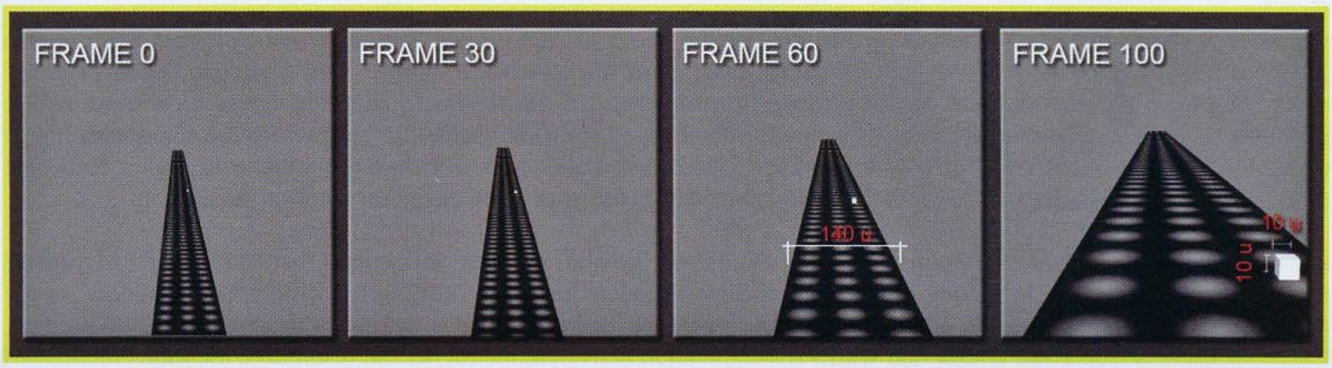
Analyzing the faceted version of the triangles, it would seem that optimal ***Edge length*** values are between 4 and 10 pixels (with the ***View-dependent*** parameter activated). In fact the small circular saw-effect which is to be found half way up the cones is visible, when using value set to 4. But if we now observe the final smoothed version, ***Normal mapping***, even with high values, helps out by creating those details that ***Displacement*** alone would not manage. It is also true that, as emphasized by the black line traced on the model, at value set to 20, ***Normal mapping*** is not enough to correct the imperfect smoothening of ***Displacement***. We have however saved time and memory. It is important to remember that these images are an extreme enlargement of a model which is much larger and therefore these details which can be noticed close-up, would probably not be ever noticed. Finally, the size of triangles is extremely sensitive and default settings are not always the best choice. ***Displacement maps*** with more details require more numerous subdivisions, whereas with softer maps one can use higher values.

View-dependent - when this parameter is active, ***Edge length*** is calculated in pixels. Value 1.0 means that the longest side of ach triangle measures 1 pixel on the screen. If disabled, the longest side of triangles is measured in ***World*** units. If we are working in cms, value 1.0 means that the length of the side of microtriangles is 1 cm long.
The scene used is made up of 2 elements only: a geometrical strip to which the ***Displacement*** map used in previous tests has been applied. The camera is animated in a span of 100 frames, and carries out a straight path.

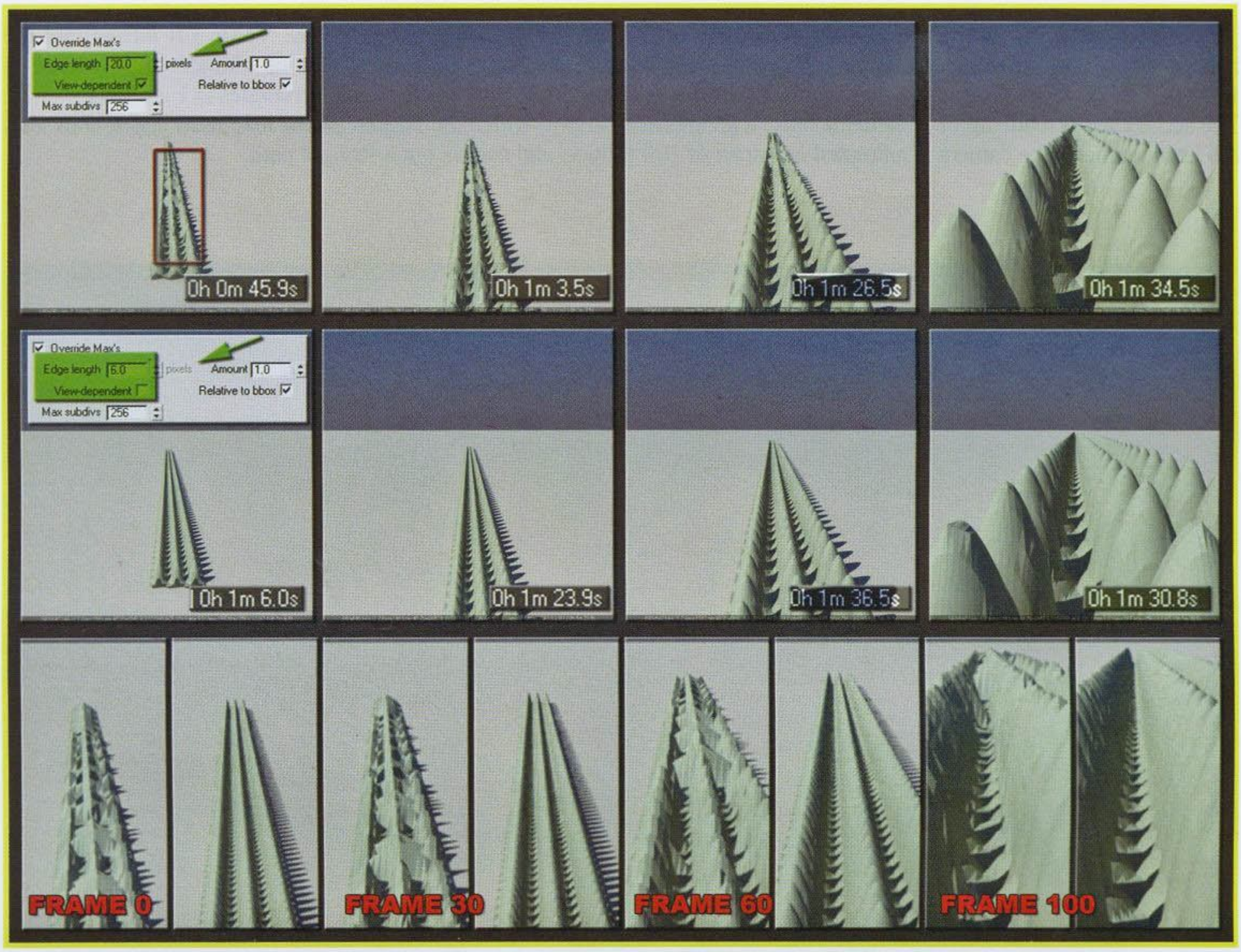


■ **Figure 4.254**
Animation scheme used for the following tests.

The width of the strip is 140 units. At the centre a small cube has been placed as a reference point. This allows one to have a first point of reference on which to base and set ***Edge length*** values when the ***View-dependent*** parameter is disabled. This way we have a first estimate of the size to use for microtriangle sides.



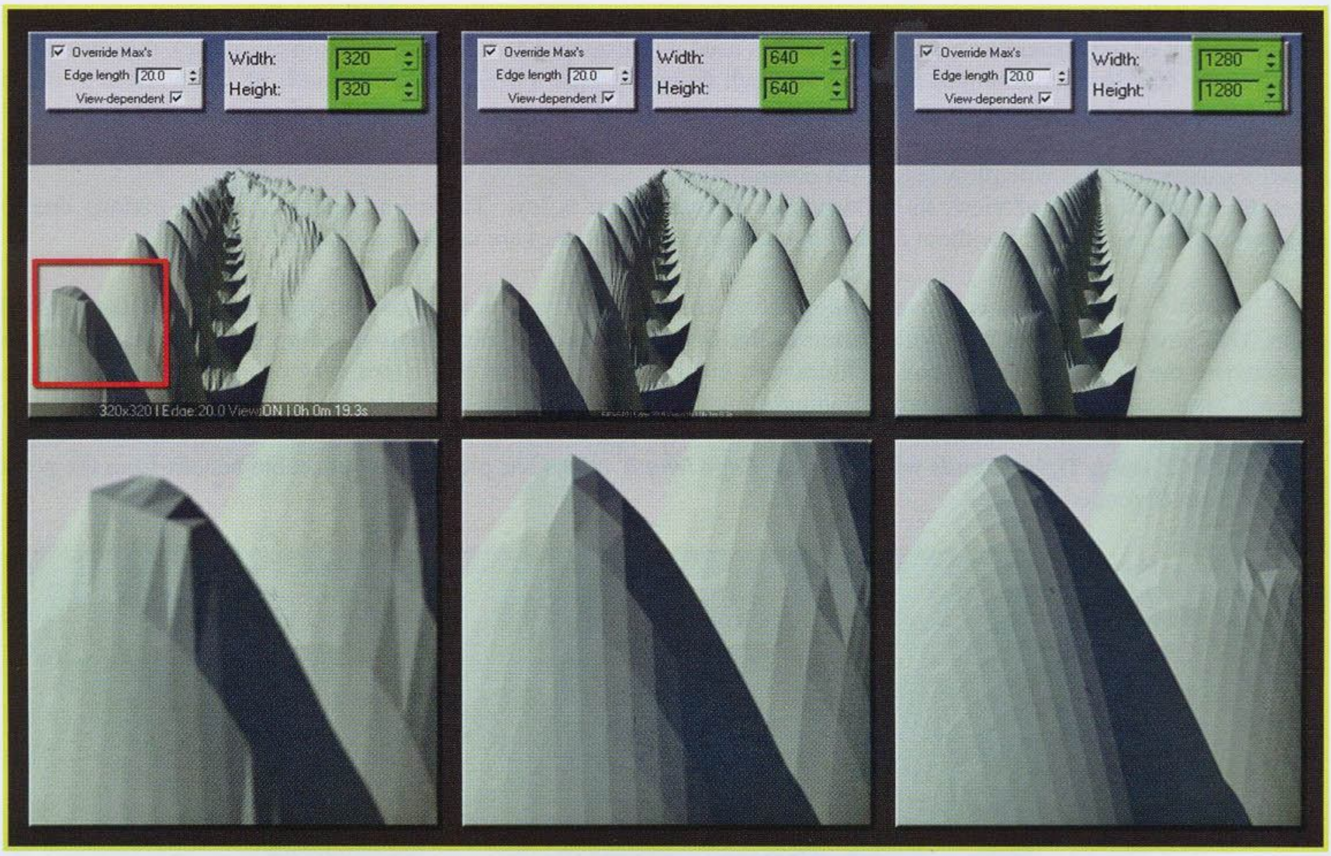
■ **Figure 4.255**
Some frames watched through the camera. ***Displacement*** is not visible in the viewport and one can only see the map that generates it.



■ **Figure 4.256**
Different results obtained by activating and deactivating the ***View-dependent*** parameter.
--- DVD ANIMATION ---

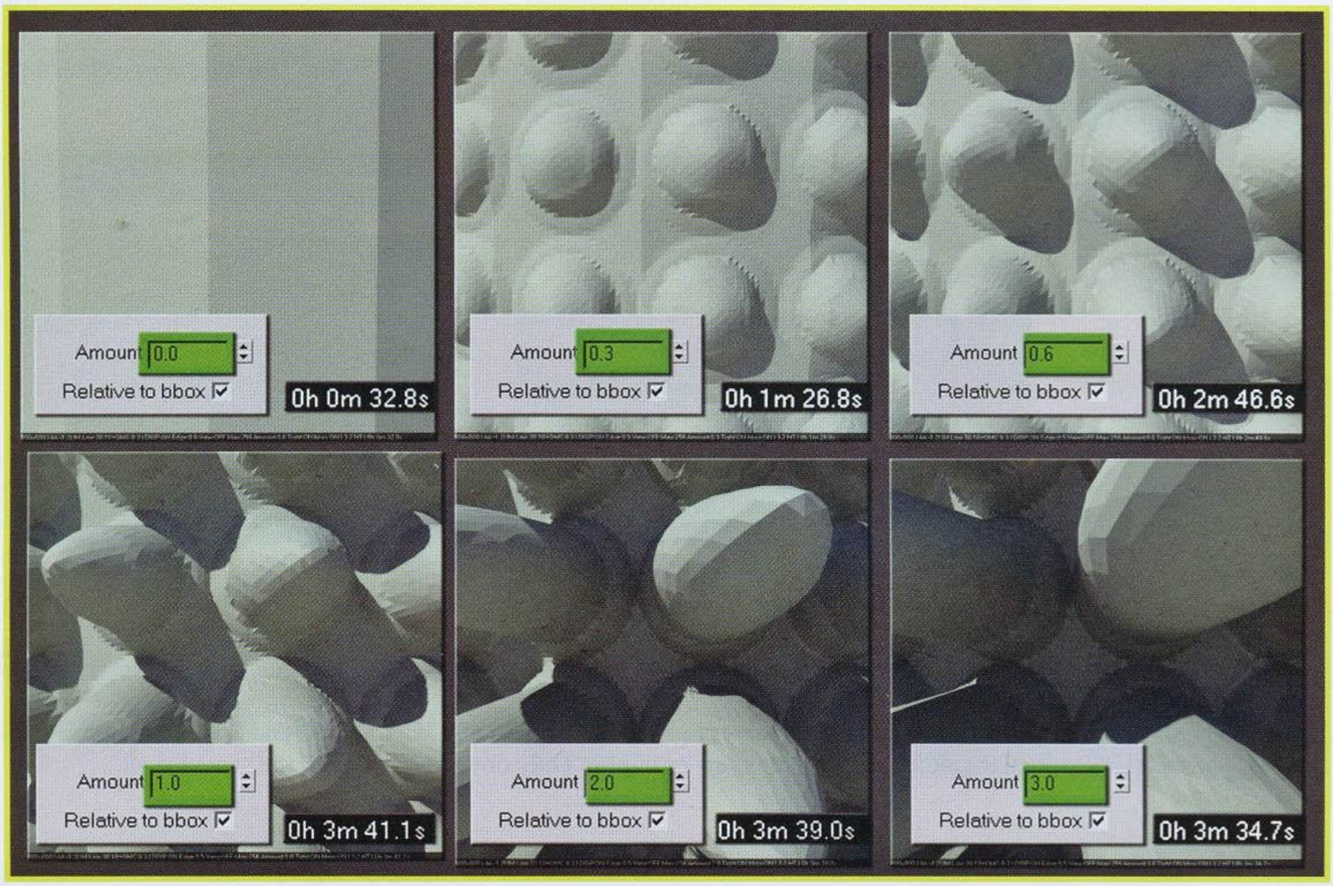
Differences are quite evident. With the ***View-dependent*** parameter activated, ***VRay*** attempts to maintain the same size for all ***Displacement*** triangles, regardless of their distance from the observation point. In this case, as the model is very long, far away polygons are less tiled then the nearer ones. This helps maintain uniformity between all triangles. Obviously in an animation this process becomes dynamic. For each frame the subdivision is actively changed. Polygons which were far away from the camera earlier on and had a low subdivision, become smaller as they draw closer to the camera. This can cause problems in animations, such as vibration effects or types of flickering.
With the ***View-dependent*** option disabled, in the rollout the word pixel is opaque. This reminds us that from now on the unit used by ***Displace*** will no longer be pixels. The images speak for themselves. The subdivision of meshes is uniform and does not suffer in the least because of the distance from the observation point. This uniformity remains in the animation too, where the triangles do not undergo transformation, if not minimally and this makes for a clean render with no artefacts. The only negative aspect is that further away many subdivisions are used which are sometimes not be necessary, because of the small size of the render. Much more memory is therefore required, with higher rendering times, as can be seen in the first frames of the animation: this is the price to be payed for a correct animation.

Also, if the ***View-dependent*** option is active, the render resolution itself influences the level of ***Displacement*** microtriangulation. In the following image three different renders have been fulfilled at different resolutions. As the measure is in pixels, as pixels increase, triangles must be more numerous in order to remain the same size. Consequently, at equal ***Edge length*** and with ***View-dependent*** activated, the higher the resolution, the more subdivided the ***Displacement*** mesh is.



■ **Figure 4.257**
Different levels of subdivision caused by the increase in output size.

Amount - this defines the amount of ***Displacement*** applied to geometry. With value 0.0 no ***Displacement*** is applied. Values above 1 amplify the effect. Negative values invert it.



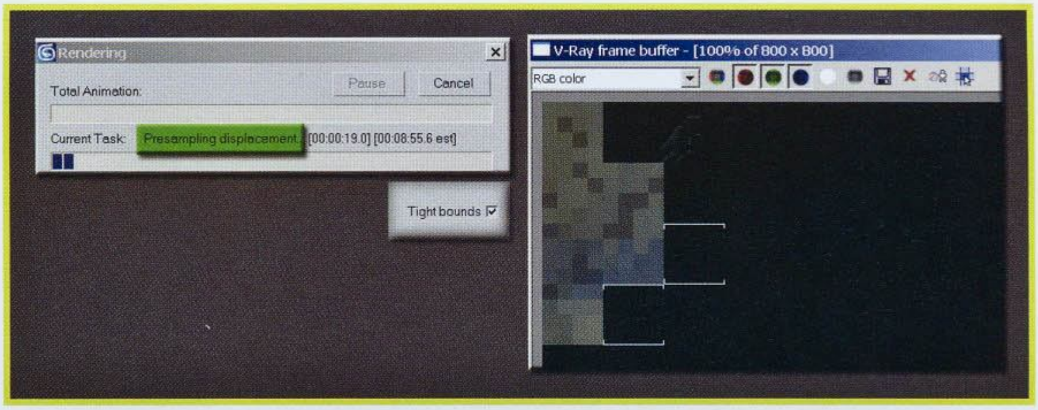
■ **Figure 4.258**
Increasing the ***Amount*** parameter, ***Displacement*** intensity also increases.

Max. subdivs - this controls the maximum number of sub-triangles generated by ***Displace***. This parameter represents the square root of the actual number of sub-triangles used. For example, if it is set to 256 (default value), this means the 256 x 256 = 65,536 microtriangles are used for each triangle which makes up the 3D mesh. It is advisable to avoid using excessive values. If one wants to achieve a greater detail, one can subdivide the original mesh, by means of modifiers such as ***Tessellate, Subdive*** or ***Turbosmooth***, instead of increasing the number of triangles with the ***Max Subdivs*** parameter. This will allow less RAM consumption.
From the practical point of view, its influence on ***Displacement*** is low. The main parameter which permits one to manage microtriangulation is without doubt ***Edge length***. Default values are usually satisfactory.

Tight bounds - when this is active, ***VRay*** tries to compute the exact volume of the ***Displacement*** of the original mesh. This however requires a pre-calculation of the ***Displacement*** map, but it will speed up rendering, especially in cases where the ***Displacement*** texture includes vast black and white areas and is not very detailed. If the difference between black and white areas is high, meaning a high level of detail, pre-calculation is very slow: in this case it is better not to activate this parameter.
In the example, using ***Tight bounds*** with very low ***Edge length*** values has provided a noticeable benefit from the point of view of saved rendering time.

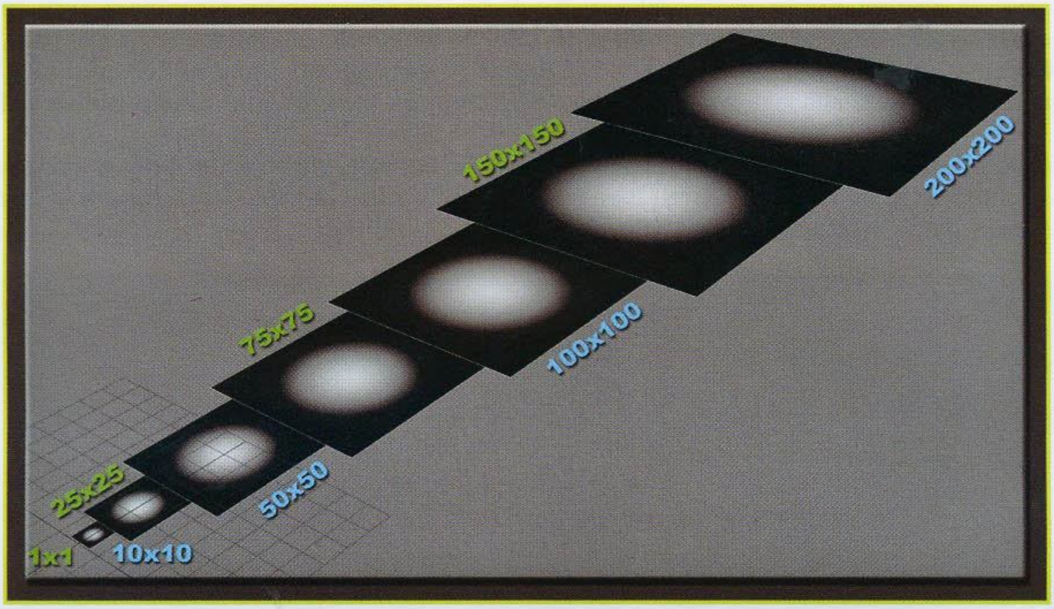
■ **Figure 4.259**
By deactivating ***Tight bounds***, the difference in time is very high.

During rendering phase, one can observe how ***VRay*** carries out pre-emptive sampling before ***Displacement*** calculation, just like during rendering.



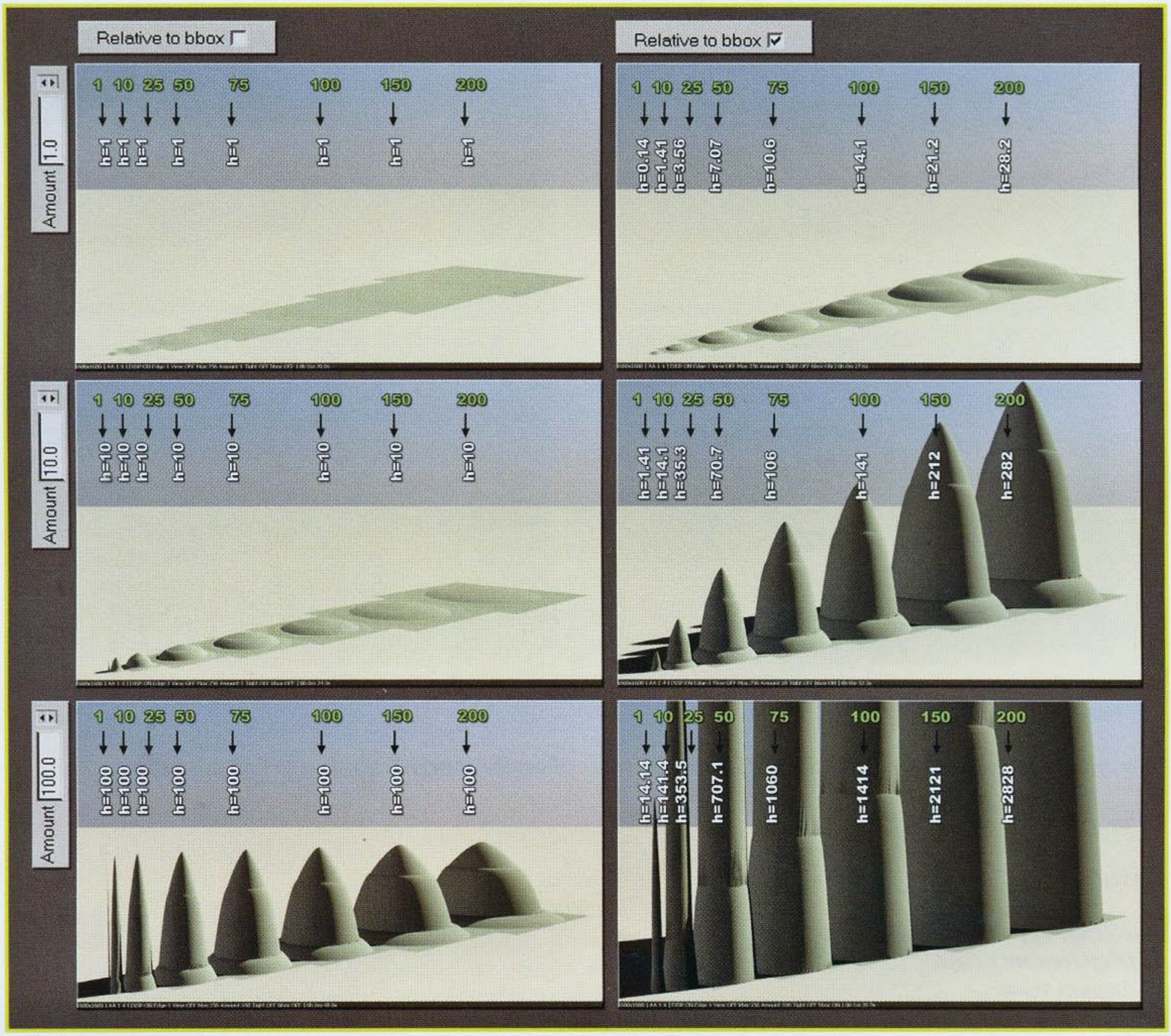
■ **Figure 4.260**
Sampling of ***Displacement*** during the pre-rendering phase.

Relative to bbox - this parameter allows one to decide whether the ***Amount*** value should be referred to the ***Bounding box*** of the object which ***Displacement*** is applied to or instead to ***world units*** being used.
This is not a very easy concept to explain and an example is therefore given in order to clear up any doubts on this rather unique parameter.
The scene is made up of seven square planes of progressive sizes: 1, 10, 25 50, 75, 100, 150 and 200. The texture used for ***Displacement*** is the circular gradient used in previous examples. Only one shader is used, the same for all.



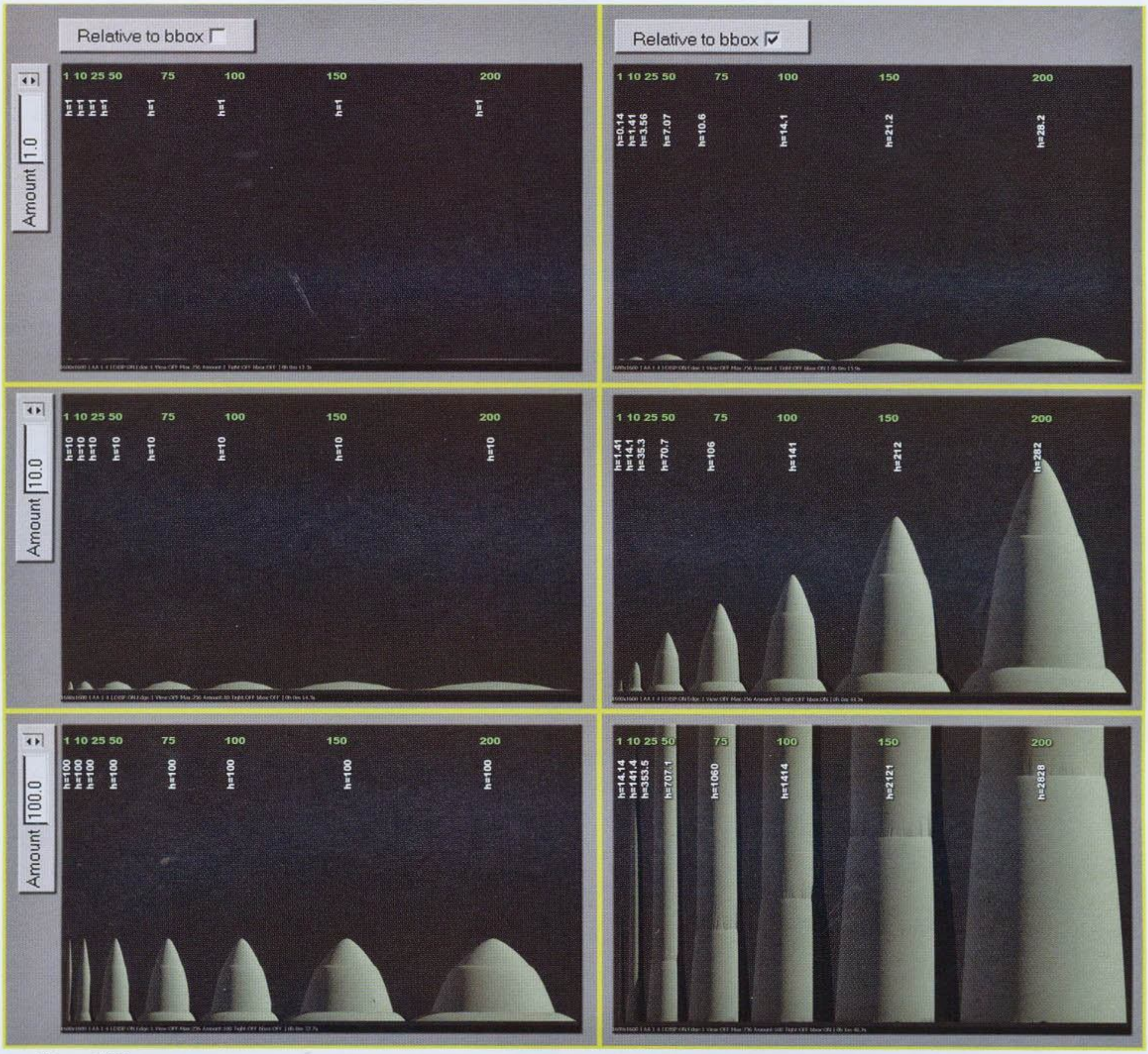
■ **Figure 4.261**
Scene used as an explanation for the ***bbox*** parameter.

The following image represents the effect of the ***Relative to bbox*** parameter. The column of images on the left contains renders where the parameter is deactivated. On the right instead, ***Relative to bbox*** is active.



■ **Figure 4.262**
increasing the ***Amount*** parameter, the intensity of ***Displacement*** is greater.

By disabling this, the meaning of ***Amount*** is quite clear: it refers to the distance of ***Displacement*** measured in units. In this case, a map has been used where black and white are absolute [0,255]. So in the case of ***Amount*** being 100 units, the central point of ***Displacement***, which in this case is a white 255 (centre of the gradient), becomes 100 units high. And so on for values 1 and 10. All planes, although with completely different sizes, have the same ***Displacement***, and the same identical height.
The matter is different instead when the ***Relative to bbox*** parameter is active. In this case, although the ***Amount*** value is constant, ***Displacement*** can fulfil different types of effects. It is evident that for small models, ***Displacement*** is less intense. Vice versa, as the model grows in size, ***Displacement*** increases.



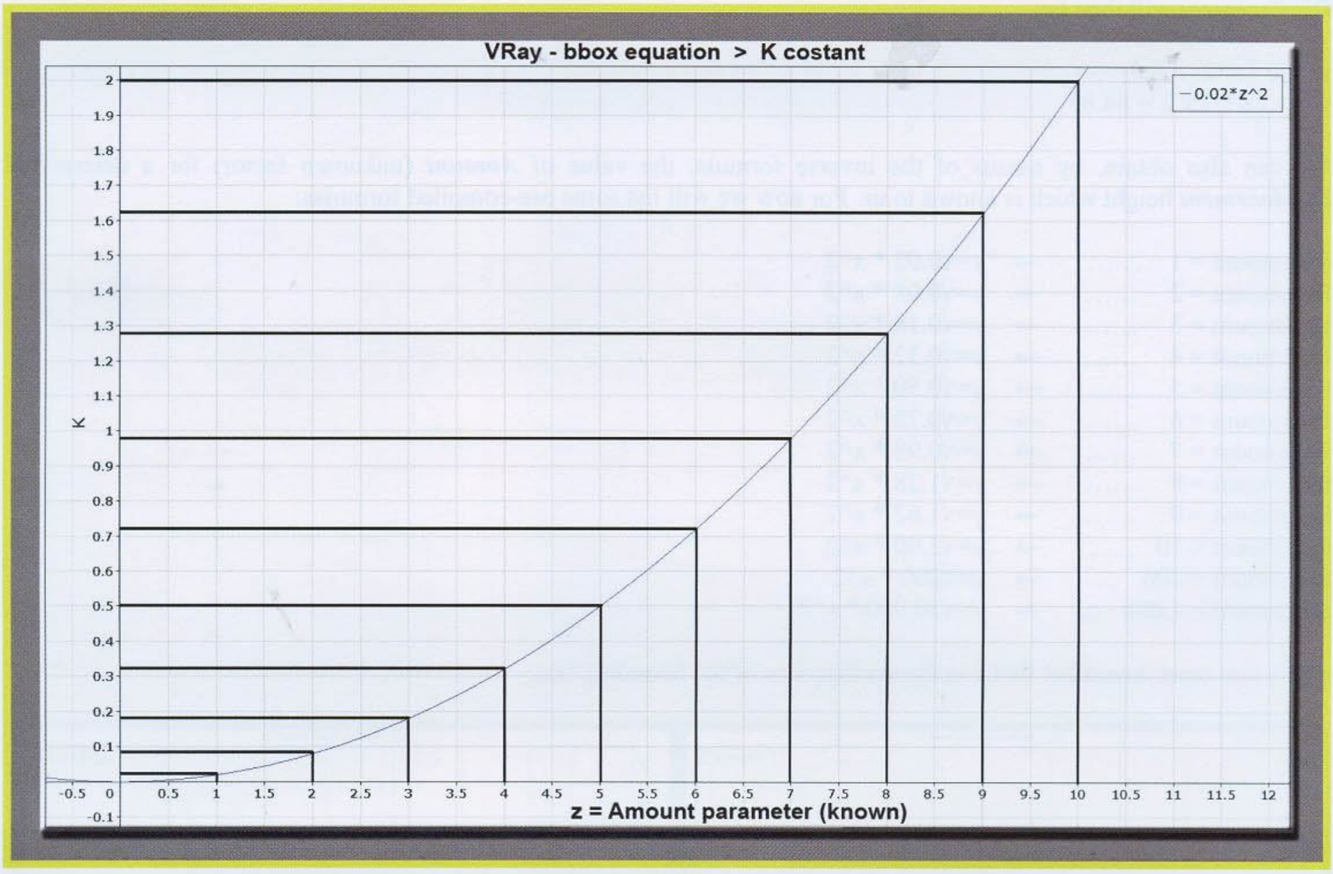
■ **Figure 4.263**
Front and orthogonal view of the test.

How can one know the height beforehand? There are a series of equations which allow us to find this value:

for amount = z → $y=\sqrt{K} * x^2$

where:

x= average size of the ***bounding box***.
y= ***Displacement*** height.
z= ***Amount*** value.
K= constant given by the following formula → $K = 0.02 * z^2$.



■ **Figure 4.264**
Diagram of the ***K*** constant.

These are a few values of K obtained via the previous formula:

for ***Amount*** = 1 ...... → K= 0.02
for ***Amount*** = 2 ...... → K= 0.08
for ***Amount*** = 3 ...... → K= 0.18
for ***Amount*** = 4 ...... → K= 0.32
for ***Amount*** = 5 ...... → K= 0.50
for ***Amount*** = 6 ...... → K= 0.72
for ***Amount*** = 7 ...... → K= 0.98
for ***Amount*** = 8 ...... → K= 1.28
for ***Amount*** = 9 ...... → K= 1.62
for ***Amount*** = 10 ...... → K= 2.00
for ***Amount*** = 100 .... → K= 200
for ***Amount*** = 1,000 ... → K= 20,000

Average size instead means the mathematical average among geometric sizes of the ***bbox*** of an object. For example a plane of 25 x 25 has a value of X equivalent to a plane of size 50 x 12.5.

A practical example has been carried out in order to vanquish any doubts. The plane in question has a size of 150x37.5, from which we can obtain the value of x.

x= [(150+37.5) / 2] = 75 (mathematical average)

We hypothesize an ***Amount*** of z = 8.

K=0.02 * z^2
K=0.02 * 8^2= 1.28

The resulting K is equivalent to 1.28.

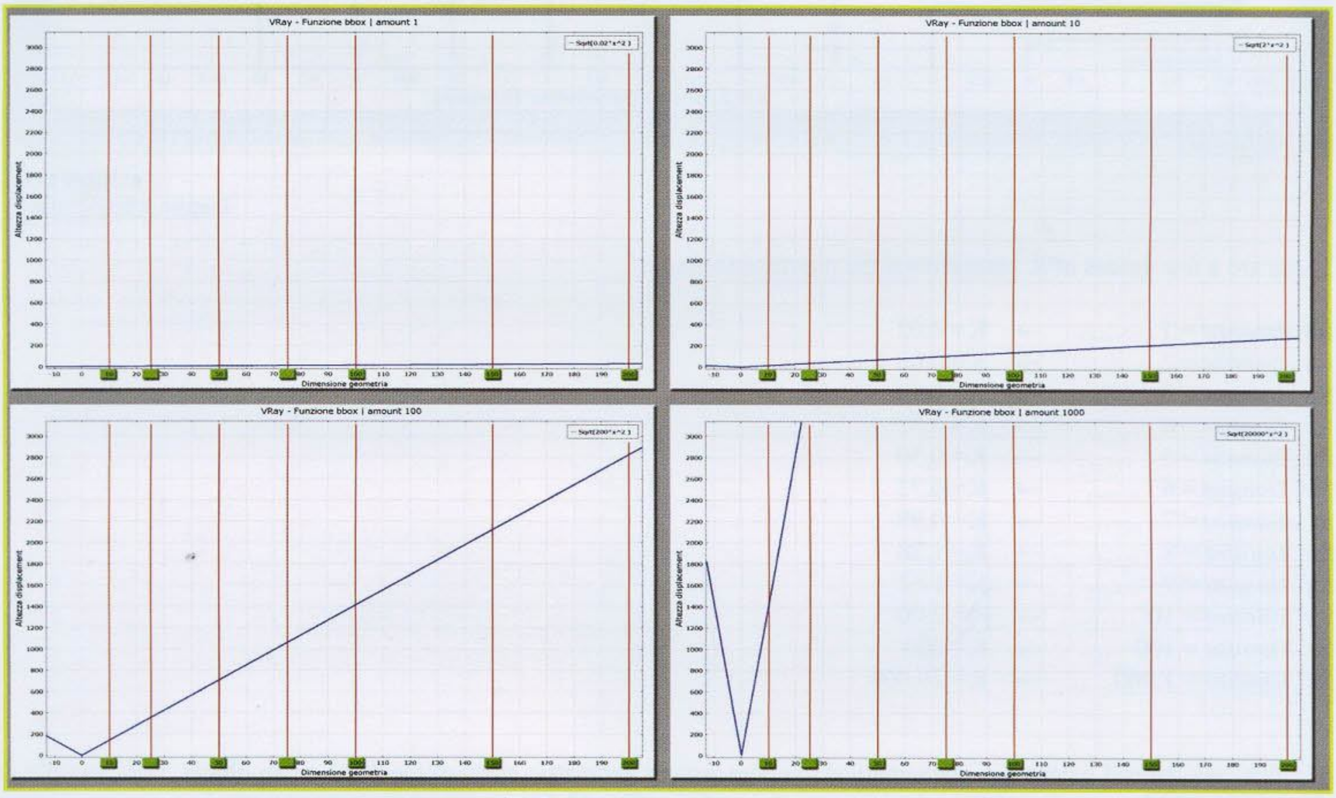
Displacement will then be:

$y=\sqrt{K * x^2}$
$y=\sqrt{1.28 * 75^2} = 84.85$

We can also obtain, by means of the inverse formula, the value of ***Amount*** (unknown factor) for a determined ***Displacement*** height which is known to us. For now we will list some pre-compiled formulas:

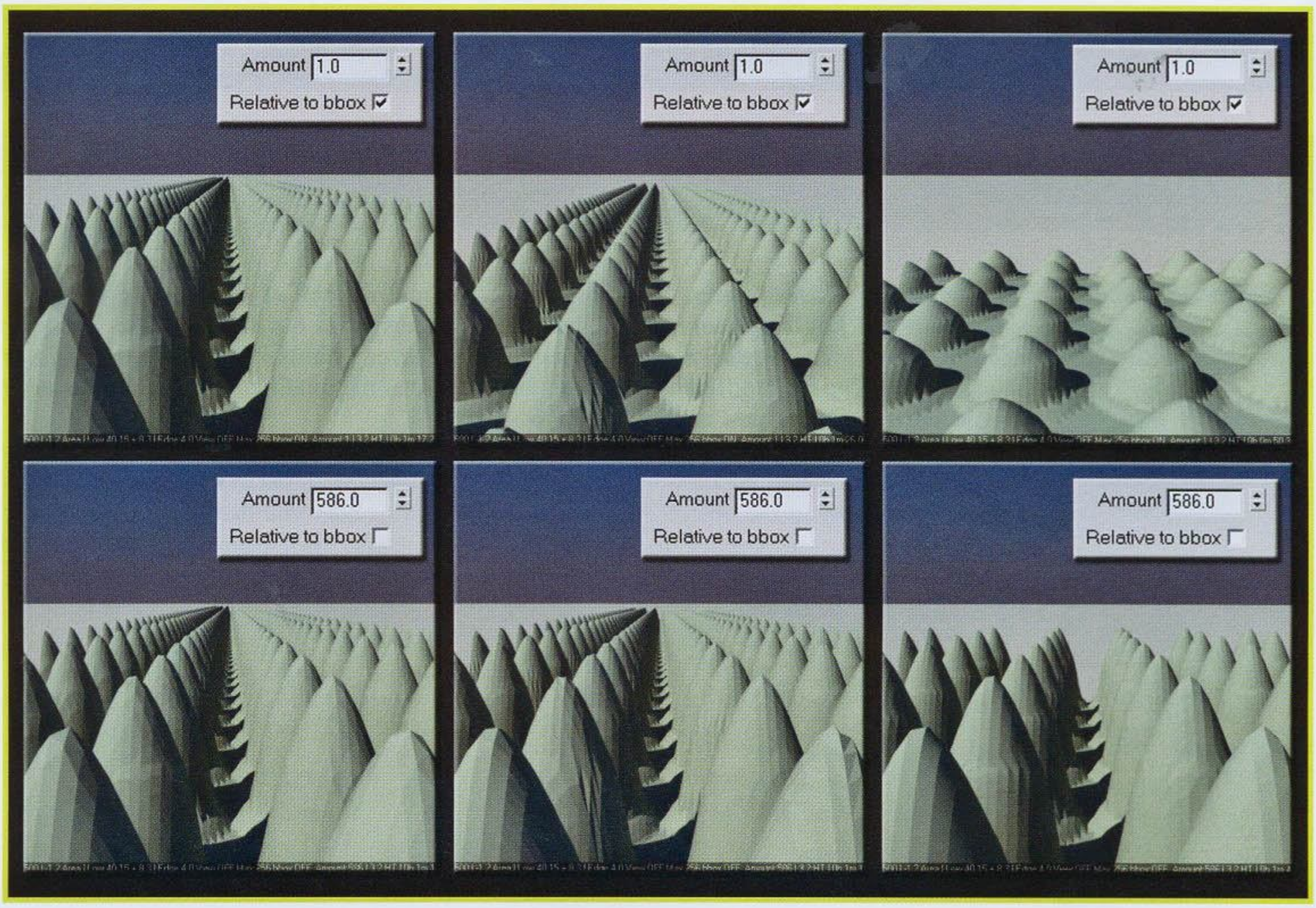
for amount = 1 ...... → $y=\sqrt{0.02 * x^2}$
for amount = 2 ...... → $y=\sqrt{0.08 * x^2}$
for amount = 3 ...... → $y=\sqrt{0.18 * x^2}$
for amount = 4 ...... → $y=\sqrt{0.32 * x^2}$
for amount = 5 ...... → $y=\sqrt{0.50 * x^2}$
for amount = 6 ...... → $y=\sqrt{0.72 * x^2}$
for amount = 7 ...... → $y=\sqrt{0.98 * x^2}$
for amount = 8 ...... → $y=\sqrt{1.28 * x^2}$
for amount = 9 ...... → $y=\sqrt{1.62 * x^2}$
for amount = 10 ...... → $y=\sqrt{2.00 * x^2}$
for amount = 100 .... → $y=\sqrt{200 * x^2}$
for amount = 1,000 ... → $y=\sqrt{20,000 * x^2}$

where one must remember that x is the average size of the bounding box.



■ **Figure 4.265**
Graphic representation of the ***bbox*** equation ($y = \sqrt{K*x^2}$) for different ***Amount*** values and therefore ***K*** as well. The x parameter is unitary.

It is obvious how important the average size of an object is according to the result of ***Displacement***. The ***Amount*** value, moreover, even if it is a fixed value, gives different results on the basis of the size of geometrical objects. At this point a problem arises: what happens when an object which has been treated with ***Displacement*** with the ***bbox*** parameter activated suffers an animation which changes its size? With a change in the size of geometries and consequently of the bounding box means that ***Displacement*** will change. Therefore we will have a dynamic displacement.
The scene used in the test is the same one used for the ***View-dependent*** parameter. The strip has however been shortened. This would cause modifications in ***Displacement*** if ***bbox*** is activated.



■ **Figure 4.266**
Geometric deformation of a mesh subjected to ***Displacement***. In the test with ***bbox*** activated, at equal ***Amount*** values, for the whole duration of the animation there are variations in the intensity of ***Displacement***.
--- DVD ANIMATION ---

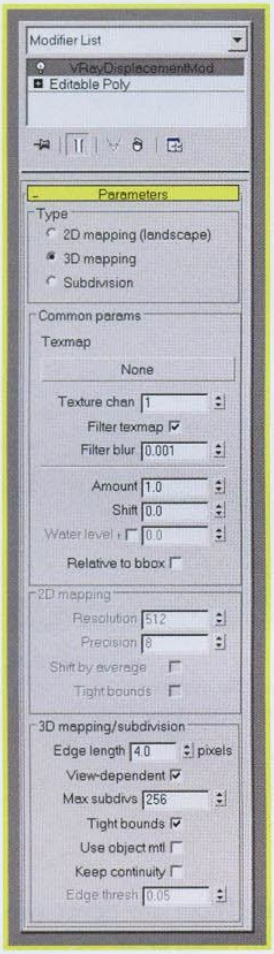
Analyzing this animation we can make two observations:

- Microtriangulation is regenerated at each frame, both with ***bbox*** enabled or disabled. There are never two frames with the same, identical subdivision. This could cause flickering problems if subdivision is set too low.
- As expected, if the ***bbox*** parameter is active, the intensity of ***Displacement*** undergoes changes in height.

In some situations one may want ***Displacement*** to change its intensity according to geometrical variations of the mesh; if this is the case, ***bbox*** should be activated. Vice versa, if one wants a fixed ***Displacement***, this parameter should be disabled.

VRay: VRayDisplacementMod

INTRODUCTION



For generating ***Displacement***, ***VRay*** offers another tool as well as the one we have just analyzed: the ***VRayDisplacementMod*** modifier. Although it is not found in the Renderer window, the ***VRayDisplacementMod*** modifier behaves similarly to ***VRay: Default displacement.*** The main difference between the two is that the modifier brings ***Displacement*** to object level and adds new and interesting features.
When one uses ***Displacement*** via the rollout in the Renderer section, ***Displacement*** is controlled by the shader which is being applied to the object. With the modifier this control is transferred directly to the object. As we will see soon, it is also true that the modifier's use permits one to pass the control of ***Displace*** over to the shader, as happens in ***VRay: Default Displacement***. In any case the ***VRayDisplacementMod*** offers a greater number of operations, allowing greater flexibility.

■ **Figure 4.267**
VRayDisplacementMod modifier applied to an Editable Poly.

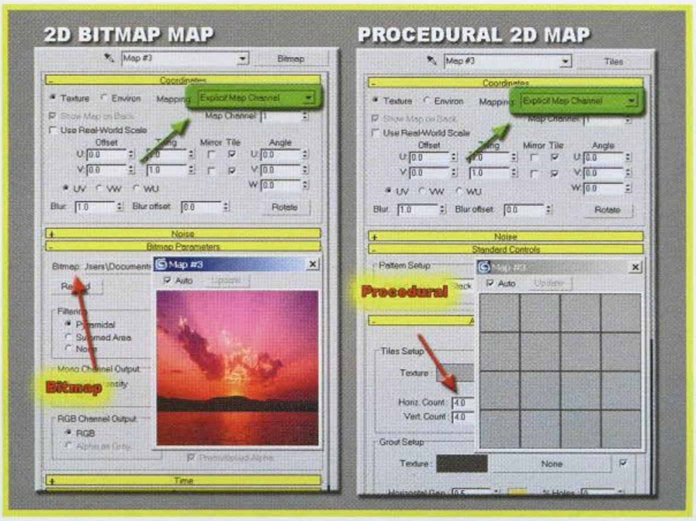
PARAMETERS

Type

Type - this defines the method used by ***VRay*** for ***Displacement***. With the ***Displacement*** modifier, as well as microtriangulation, which we have already seen, other types of calculation are offered.

2D mapping (landscape) – this method requires objects to have ***UVWMap*** coordinates. This is simply because with this system ***Displacement*** is calculated in Texture Space. The advantage of this method is to preserve all the details of the map used for ***Displacement*** within ***Displacement*** itself. On the other hand the model must have valid map coordinates. One cannot use ***2D mapping*** with procedural 3D textures or textures that use object or global coordinates, such as Cellular, Marble, Waves or Noise etc... unless they use Explicit map channel mode, having previously assigned UVW coordinates to the object. The same is true for procedural 2D maps such as Tile, Gradient o Checker. Also these must use Explicit map channel, which is in any case the default, together with UVW coordinates applied beforehand. In terms of quality, ***2D mapping*** can fulfill better ***Displacement*** effects compared to ***3D mapping***, and is also much faster.

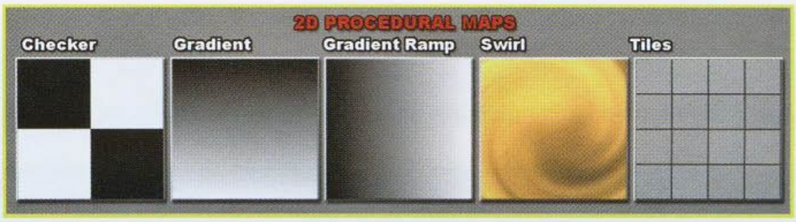
First of all we must understand the difference between 3D maps and 2D maps. 2D maps are textures used for mapping the surface of geometrical objects or also used as environmental maps for creating a background for scenes. The most simple ones are bitmaps, images captured with digital tools. 2D maps can also be procedural. This means that their appearance is due to mathematical procedures and one can change their shape, color variation, etc... by means of appropriate parameters found in the Material Editor of 3ds Max.



■ **Figure 4.268**
Examples of procedural 2D maps.

A trait of 2D maps is that by default they are in Explicit map channel mode. Their behaviour is similar to that of a decal. They allow one to "dress" objects and assign these objects with shader information. The types of 2D procedural maps are:

. Bitmap
. Checker
. Combustion
. Gradient
. Gradient Ramp
. Swirl
. Tiles



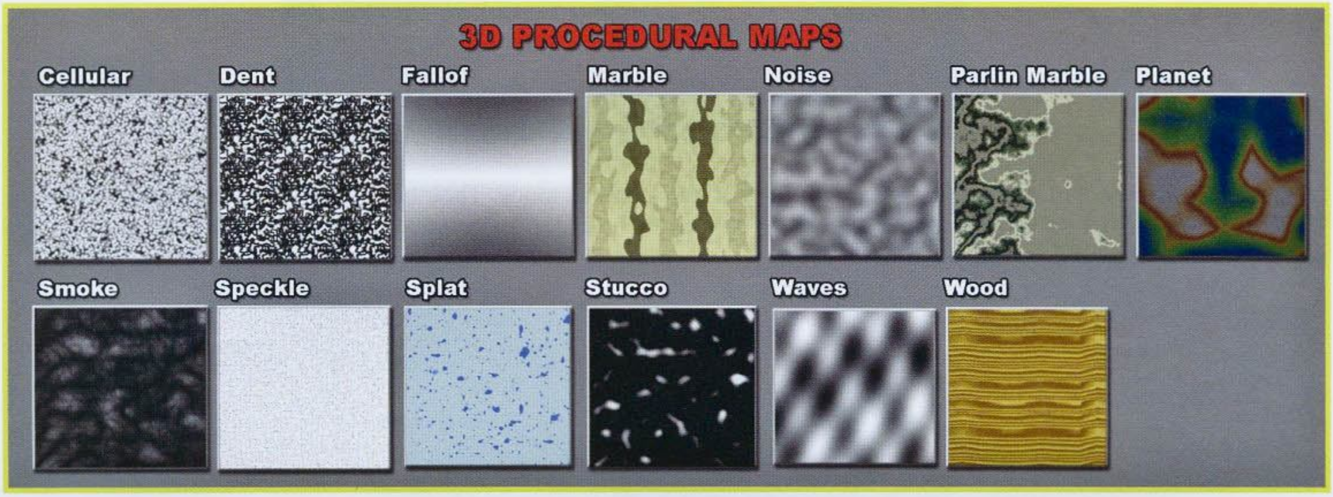
■ **Figure 4.269**
The 2D procedural maps of 3ds Max. The Combustion map is not shown as there is no preview available.

The remaining maps are 3D procedural textures. In this case their nature is mathematical, exactly like 2D procedural maps; the only difference of this type of maps is the fact that they are in three dimensions instead of two. By assigning a 3D procedural map to an object, the whole volume is included in mapping and not just the surface. Even the inside is mapped in a correct and continuous way. So, if one decided to cut a geometrical object in half, the whole object would be perfectly texturized and aligned, like a block of marble with its veins. Surface details continue correctly also on the inside.

■ **Figure 4.270**
Complex models texturized with 3D procedural maps only. None of these has made use of UVW mapping.

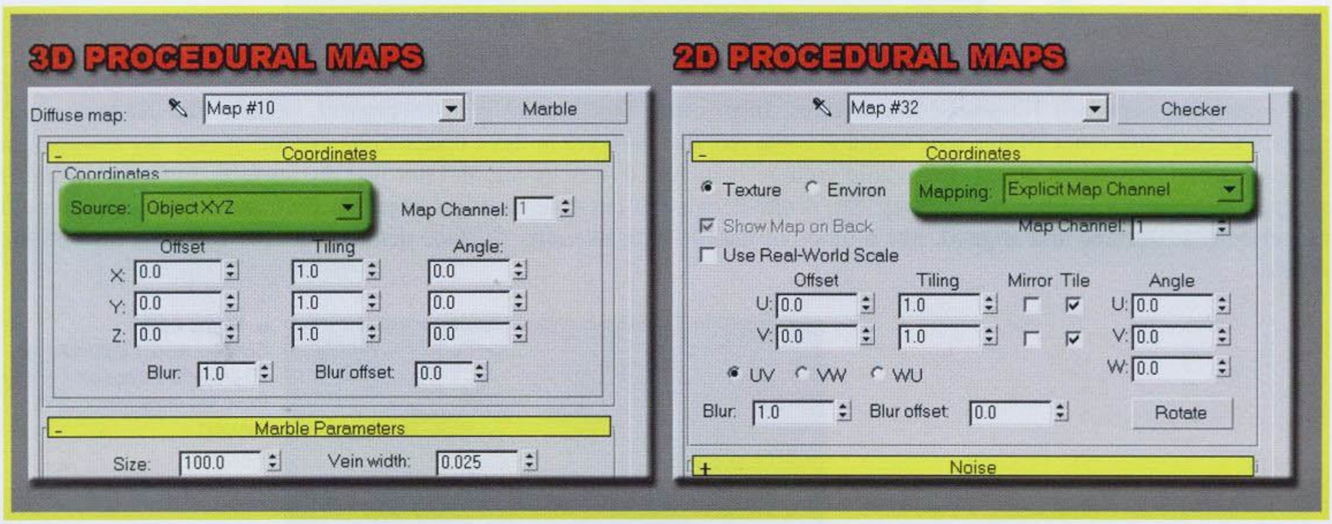
The remaining 3D maps are:

. Cellular
. Dent
. Fallof
. Marble
. Noise
. Particle Age
. Particle MBlur
. Parlin Marble
. Placet
. Smoke
. Speckle
. Splat
. Stucco
. Waves
. Wood



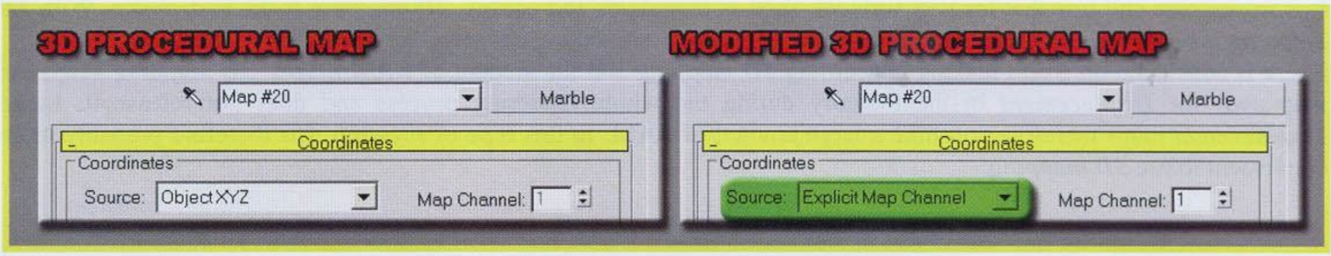
■ **Figure 4.271**
3D procedural maps in 3ds Max. The maps Particle age and Particle Mblur are not shown because the preview is not available.

The main difference between a 3D map and a 2D map lies also in mapping management.



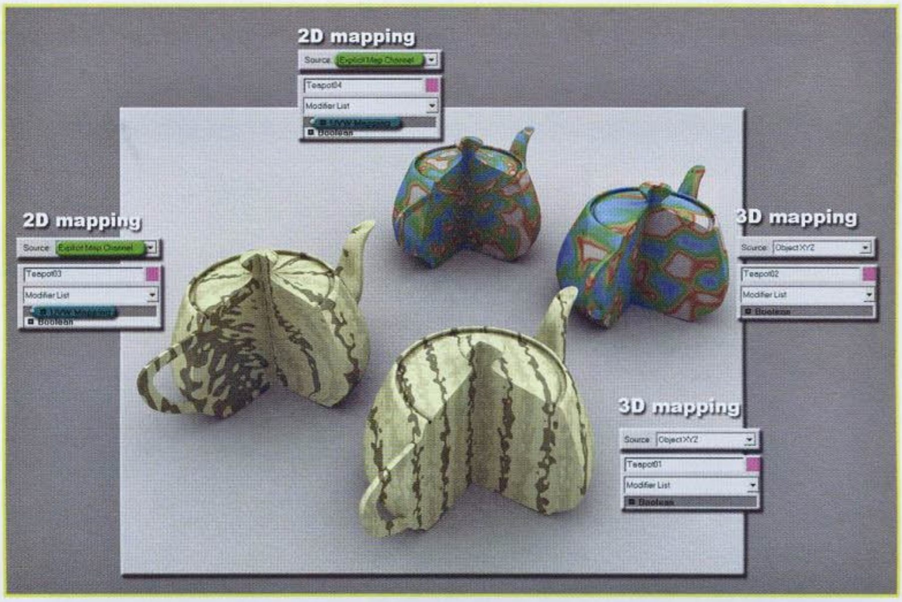
■ **Figure 4.272**
Handling of mapping coordinates within 3D maps is extremely different to that of 2D maps.

The particularity of these textures is however not perceived by the ***VRayDisplacementMod*** which needs the object to have precise UVW coordinates, in the case of use of 2D mapping. Therefore also 3D maps have to be transformed into two-dimensional textures, transferring their 3D peculiarities into 2 dimensions. In order to do this, one must change the 3D coordinates from Object XYZ (default) to Explicit Map Channel.



■ **Figure 4.273**
Modification of the system of coordinates in procedural 3D maps.

This way a 3D map become nothing less than a 2D map. At this point however there must be valid UVW coordinates for the mesh to be correctly texturixed. This can be obtained by using the UVW Map or Unwrap UVW.



■ **Figure 4.274**
Procedural maps in Explicit Map Channel mode and Object XYZ.

Notice the fact that in models where the 3D map has been applied in ***Explicit map channel*** mode, internal continuity no longer exists in objects. For this reason one has to resort to spherical UVW coordinates.

This long introduction to mapping systems in 3ds Max is useful to understand the difference between 2D and 3D maps. The ***2D Displace*** in ***VRay*** can use 2D maps, procedural or not and also 3D maps. The latter must be modified with technique we have just described.

3D mapping - this method is similar to ***VRay:Default displacement*** in the Renderer panel of ***VRay***. In fact it subdivides the mesh into microtriangles which are then used for ***Displacement***. Any map can be used, 2D or 3D. It is therefore possible to exploit the procedural quality of 3D maps. This happens because with ***3D mapping*** one can maintain a continuous ***Displacement*** also along the edges of geometrical objects and areas which are difficult for ***2D mapping***.
Moreover, with this method, ***Displacement*** maps can be used, using 3ds Max maps as well as those belonging to the modifier. Therefore one can assign more than one ***Displacement*** map to the same object, making use of the IDs of faces and the Multi/sub-Object shaders.

Subdivision - this method is similar to the ***3D mapping*** system, with a difference. During the rendering phase, a subdivision scheme is applied to it, which applies an effect similar to the Turbosmooth modifier. If one wanted to apply only the ***Turbosmooth*** effect to the object, it would be enough to reset the ***Amount*** parameter. This way map ***Displacement*** is not rendered, but the ***Turbosmooth*** effect is present.

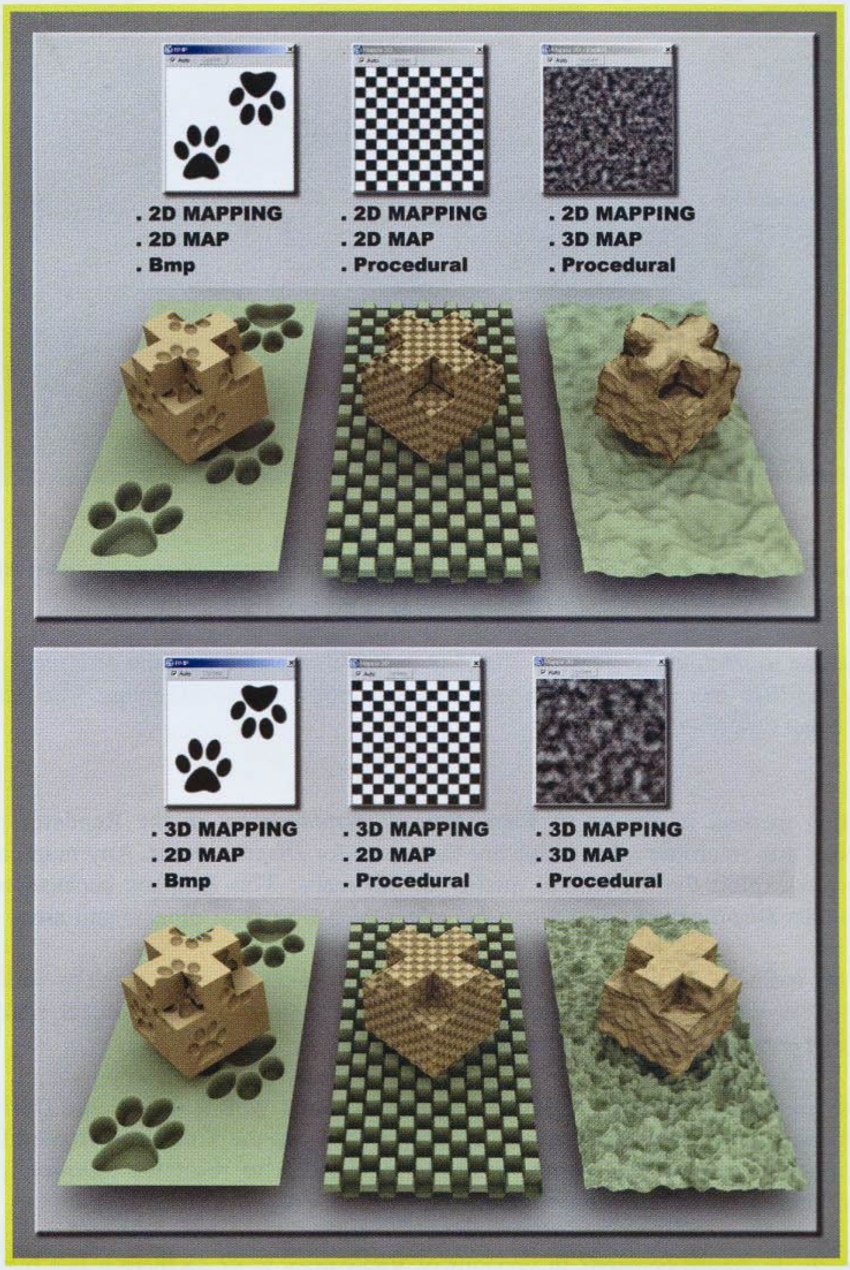
Which of these methods should be used? In versions of VRay prior to ***v1.45*** there was a big difference in performance between 2D and 3D methods and in most cases 2D mapping was the best choice for results and time-consumption. With the introduction of dynamic memory in ***VRay 1.45***, the ***3D mapping*** method has become much faster and with similar, if not better qualities, (it depends on the case and the maps used) than ***2D mapping*** systems.

It is in any case still true that for vast surfaces, such as mountains and oceans, ***2D mapping*** is the best choice. With ***3D mapping*** instead, one can attain continuity between surfaces, something ***2D mapping*** is not able to fulfil.
There is consideration worth making concerning ***2D mapping*** and the amount of RAM it uses. ***2D mapping*** keeps the map used for ***Displacement*** in the memory during the whole rendering process and does not release it. Large ***Displacement*** maps therefore consume a large amount of RAM. In this type of situation it is therefore probably more convenient to use ***3D mapping***.

Common params

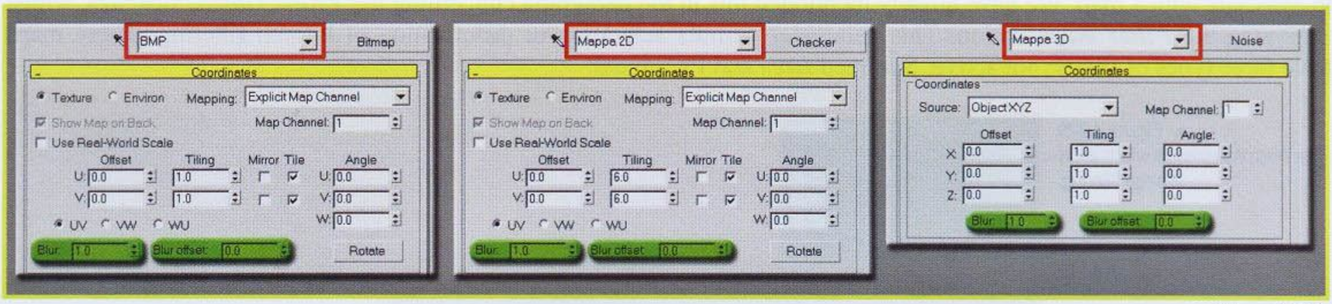
Texmap - this represents the map for ***Displacement***, both in ***2D*** mode and in ***3D mapping***. Any type of map can be used: Bitmap, procedural, 2D or 3D. With ***2D mapping*** however, one can only use maps with ***Explicit UV*** coordinates, whereas ***3D mapping*** does not impose any limitations. If the ***3D mapping***: ***Use object mtl*** is activated, the texture in the ***Texmap*** is not considered and the one in the Material Editor is used. Other options in common between ***2D mapping*** and ***3D mapping*** are: ***Texture channel***, ***Filter map***, ***Filter blur***, ***Amount***, ***Shift***, ***Water level*** and ***Relative to bbox.***

■ **Figure 4.275**
Usage of different types of 2D and 3D maps for ***Displacement***. The main difference between ***3D*** and ***2D mapping*** can already be noticed here: the ability to make adjacent surfaces continuous along abrupt edges.



Texture channel - this represents the UVW channel used by ***Displacement***. This value must correspond to the specified channel in the texture used for ***Displacement***. If the option ***Use object mtl*** is active, this value is not taken into account.

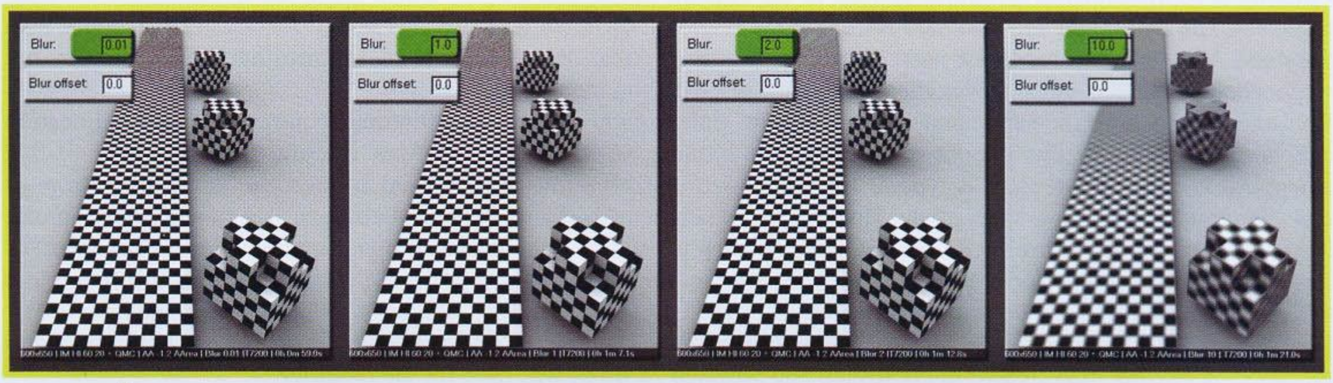
Filter map - if this is active, the texture used for ***Displacement*** will be filtered. This parameter is not considered when ***Use object mtl*** is active.
Before continuing, we will focus our attention on the concepts of ***Filter*** and ***Blur***, parameters which are available for any type of map within 3ds Max.



■ **Figure 4.276**
Blur options in the 3ds Max map.

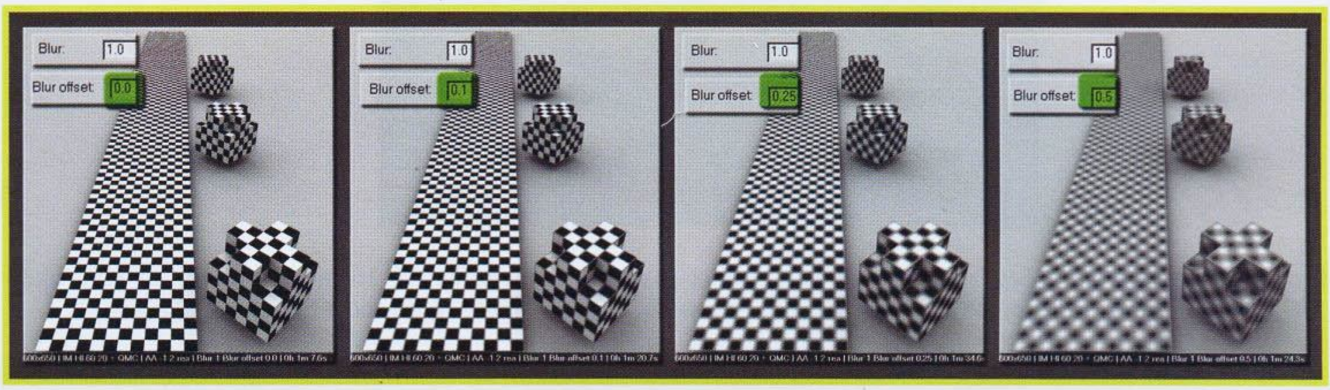
These parameters are to be found in every map and influence the Blur effect in maps. They are applied to textures which are to generate ***Displacement*** and, for this reason, it is plausible to expect them to influence the ***Displacement*** result in some way.

Blur - it acts upon definition or blurring of the map according to the distance compared to the observer's point of view. The greater the distance, the more the blurred it is. The Blur value has effect on blurring of maps in global space. Blur is mainly used to avoid aliasing in situations whereby a far away mapped object creates artefacts in the distance, because of a texture which is excessively fine. By increasing the Blur value, the pixels which are prone to this problem tend to "mix", reducing the flickering effect.



■ **Figure 4.277**
Blur parameter being used. The further the geometrical shapes are, the more the texture is blurred.

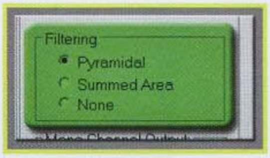
Blur offset - it takes effect on definition and blurring regardless of the distance from the observer's point of view. Blur offset defocuses the image in object space. This option can be used when one wishes to blend or blur the details of a map, in order to obtain a hazy image.



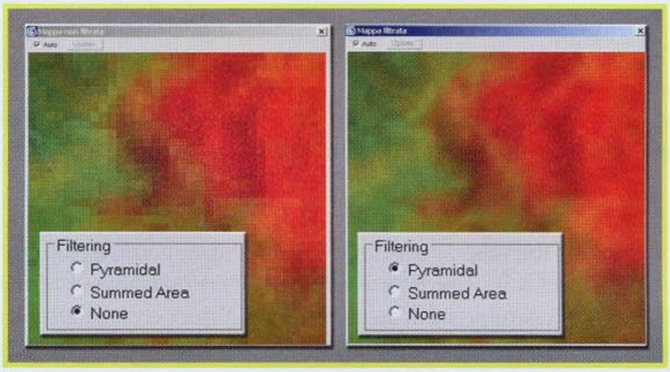
■ **Figure 4.278**
Blur offset parameter being used.

Looking at these renders, one can see how the Blur offset effect has a uniform effect on the blurring of the texture, whatever the distance from the camera. This parameter is useful when one wishes to make the ***Displacement*** smoother and reduce the sharpness of textures, due to low resolution images or excessively sharp outlines of objects.

There is also a series of parameters, this time only available for Bitmaps, by which one can choose the type of filtering to associate with a map. We have already mentioned this in the paragraph concerning the ***VRay: Global switches***, whilst analyzing the ***Filter maps*** option. This parameter permits activation or deactivation of bitmap filtering. These maps have three types of filters: not active, Summed Area and Pyramidal.

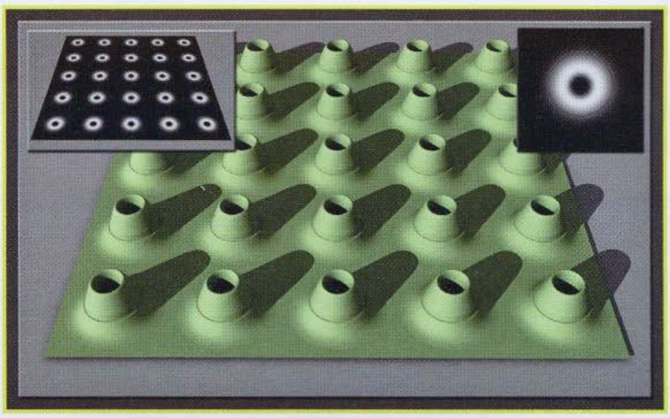


■ **Figure 4.279**
The three parameters which allow Bitmap filtering.



■ **Figure 4.280**
Textures with the Pyramid filter active and inactive.

Filtering is an ***antialiasing*** technique applied to the bitmaps of mapped shaders which make use of chromatic levelling of pixels. The Pyramidal and Summed Area options offer two different methods. Pyramidal requires less memory than Summed Area and is suitable for most situations. Summed Area, on the other hand, requires additional memory but returns better results. None disables filtering, drastically reducing memory demands.
Both methods require the same amount of rendering time, roughly. Pyramidal filtering requires a memory allocation equivalent to 133% of the amount required by the bitmap, whereas Summed Area filtering requires 400% of that amount. It is advisable to use this type of filtering only for small bitmaps to be used in the scene moderately. In most cases Pyramidal filtering is appropriate. However, as this type of filtering applies its changes as a function of distance, there may be irregular stair-effects on the detailed maps it is applied, for instance, to a plane moving away. The effect of this filtering on such perspectives is even more visible in animations, where parts of the map may seem to move in a wave-like fashion. If this happens, it is advisable to activate the Summed Area filtering for that type of shader.
For the tests, a very simple scene has been used: a plane and a direct light. ***VRayDisplacementMod*** has been applied to the plane. A circular map created in Photoshop and repeated 25 times was used for the modifier.

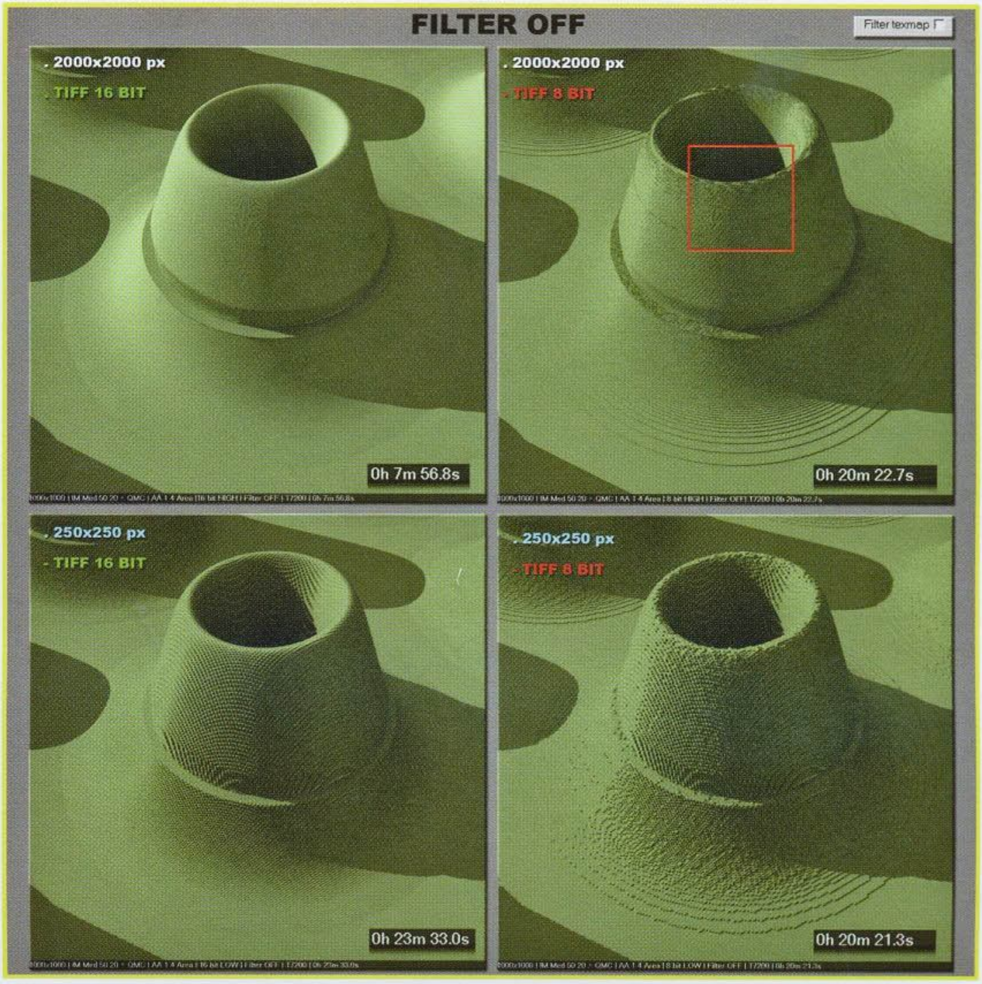


■ **Figure 4.281**
Plane modelled via ***Displacement***.

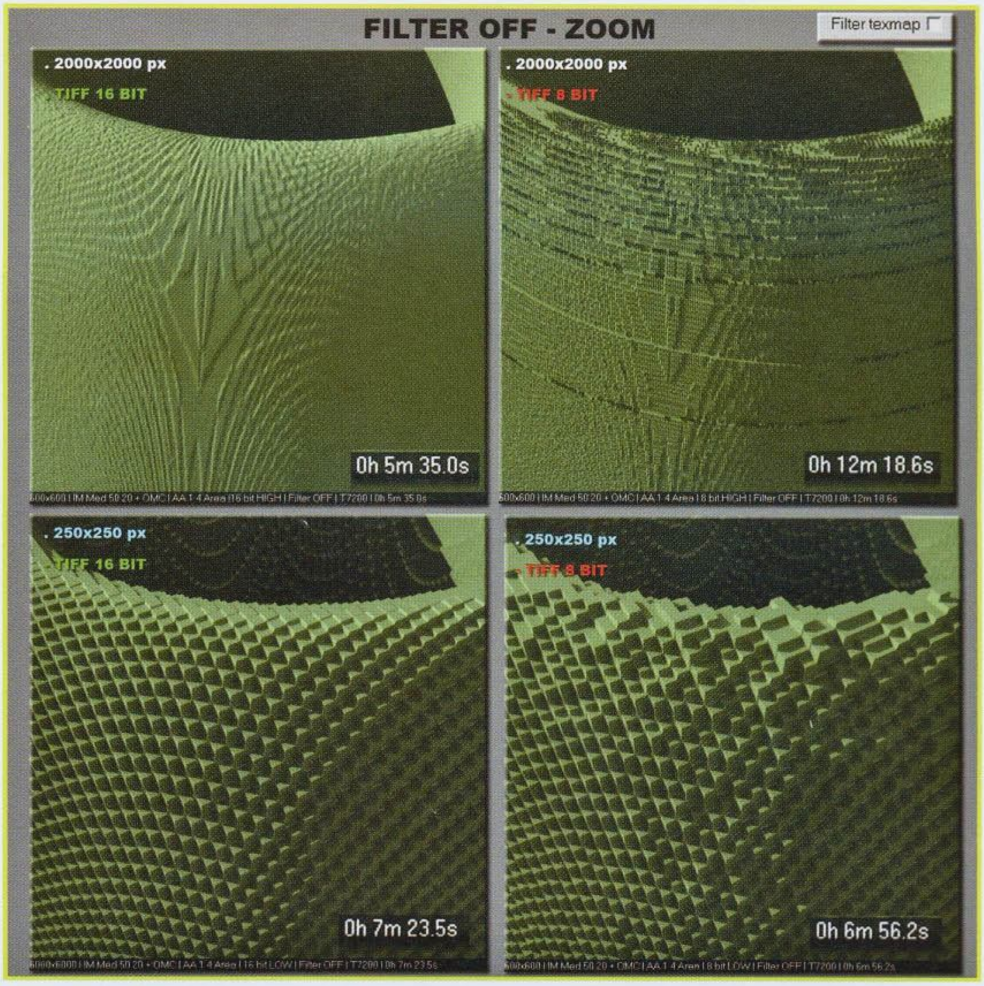
The textures which have been used are of four formats: one in high resolution, 16 bit in a range of grey tones (2000x2000 pixels), one in high resolution but 8 bit and two in low resolution, 16 bit and 8 bit (250x250 pixels).

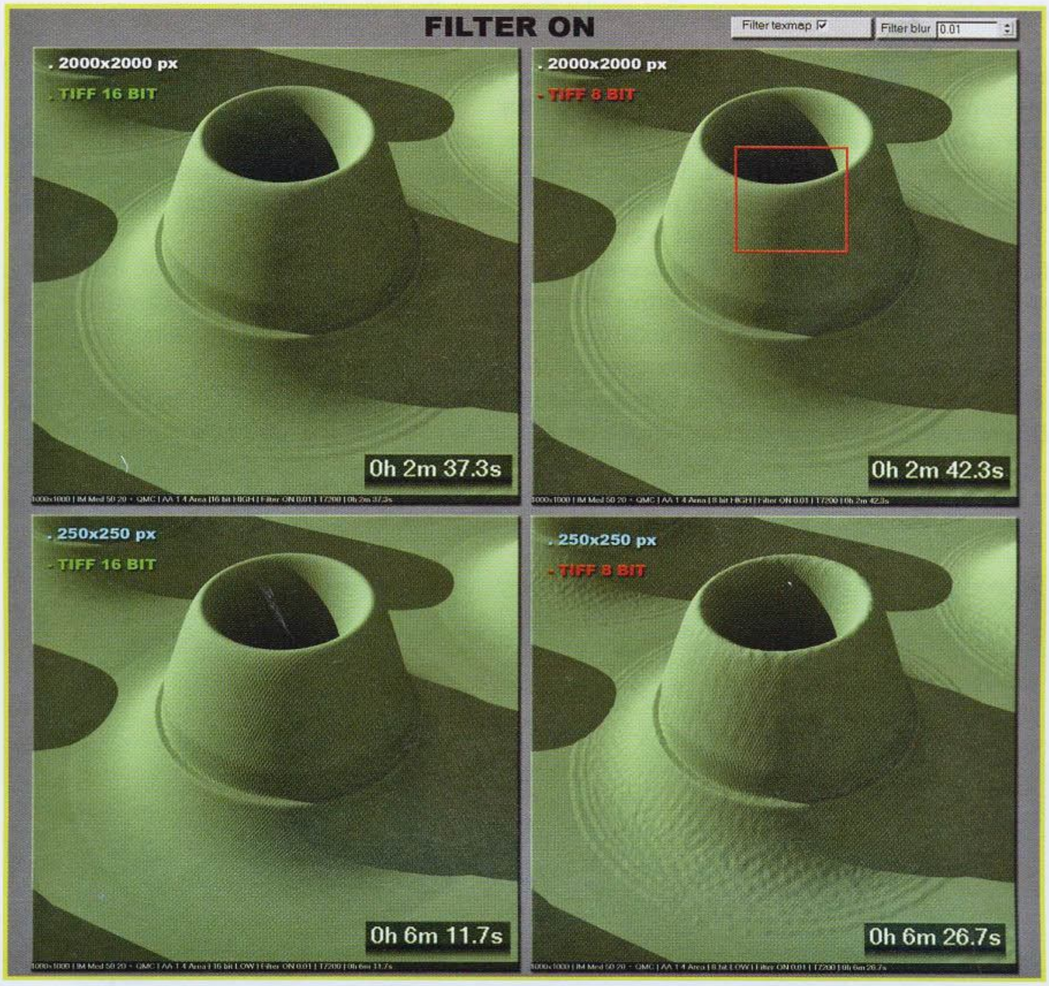
--- THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK ---

■ **Figure 4.282** Different results acquired using maps for ***Displace*** at different resolutions and color depths. No filtering has been applied.

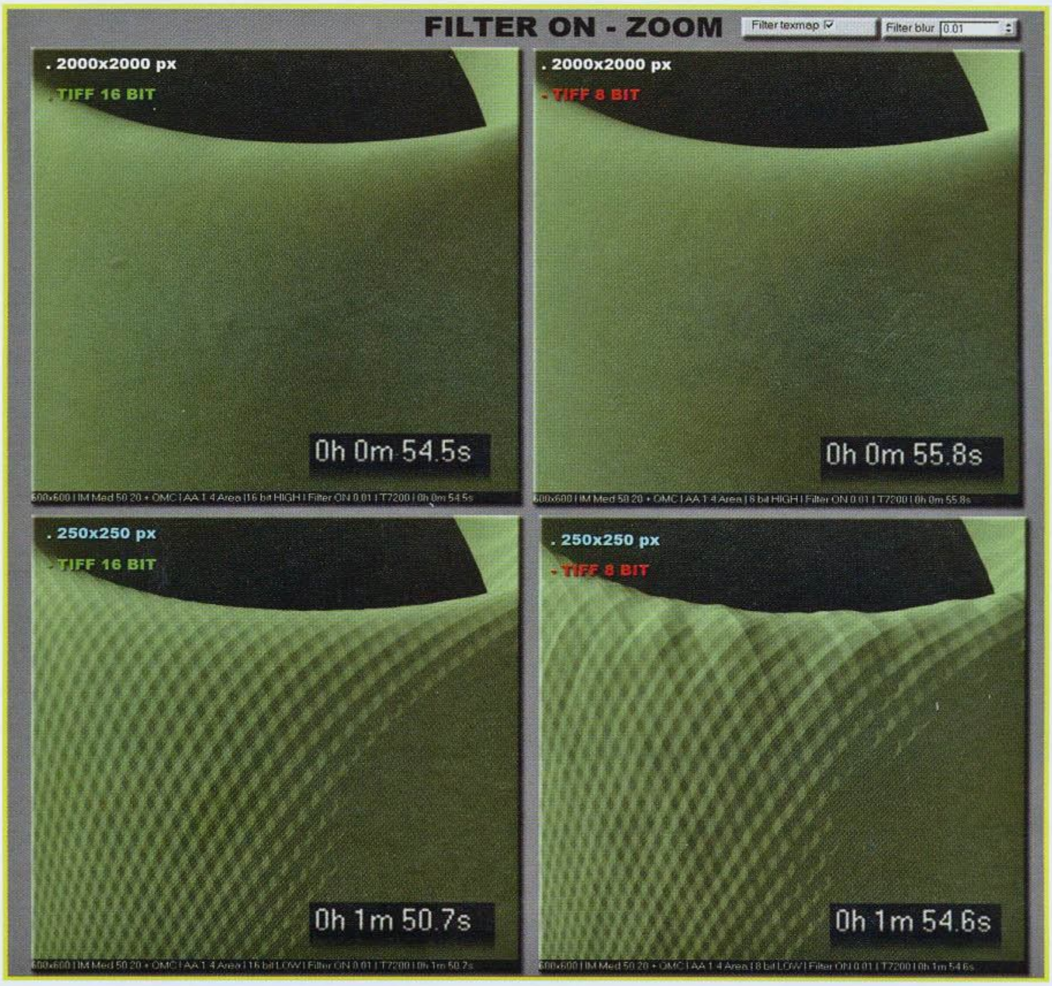


■ **Figure 4.283** Close up details of ***Displacement***. The single pixels which form the map can be seen.





■ **Figure 4.284**
Different results acquired using maps for ***Displacement*** at different resolutions and color-depths. Filtering is now active.

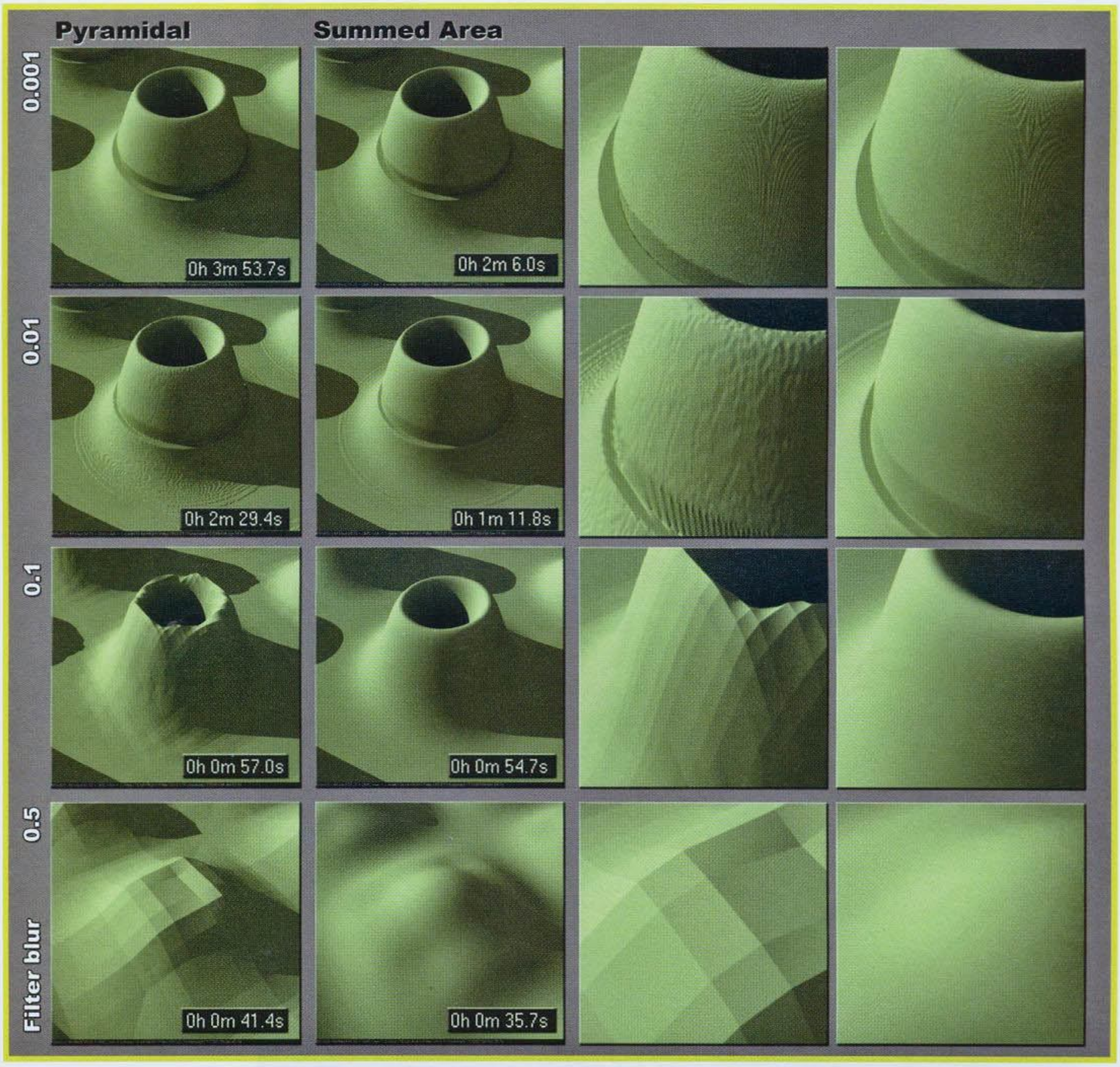


■ **Figure 4.285**
Close-up details of ***Displacement*** with active filters.

After having seen the results of these tests the conclusions that can be drawn are the following:

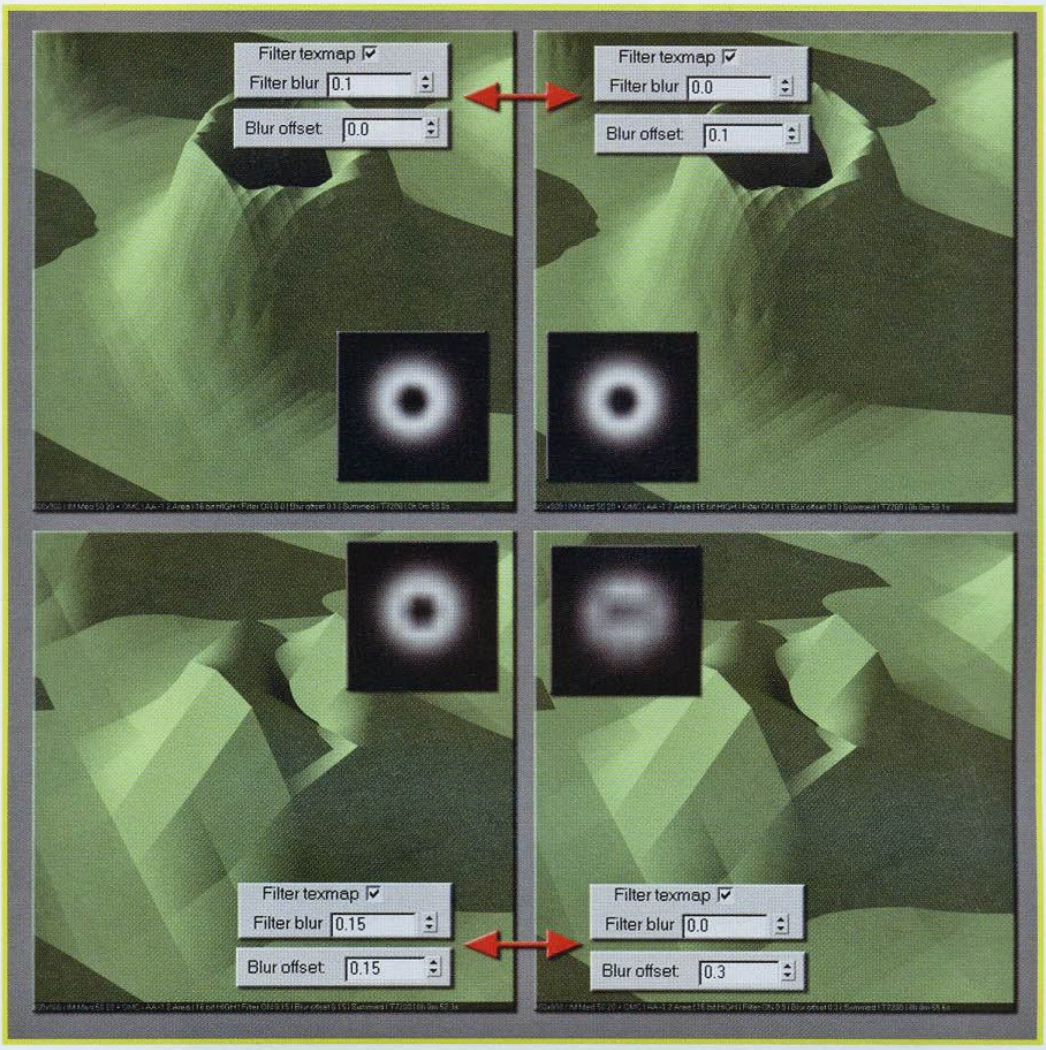
- ***Displacement*** maps, even when at high resolutions, tend to create Stair-effects. The pixels the map is made of tend to create little "steps" that from a certain distance can be seen.
- 16 bit maps produce a ***Displacement*** of significant quality and perfection. This fact can be observed by taking a look at the close-up render with filter disabled. The 16 bit image, as opposed to the 8 bit image, has produced a well-ordered geometrical pattern, which has a positive effect also on the smoother version. Instead in the high resolution 8 bit version, the messy stair-effect is evident.
- The use of filtering is a significant benefit for ***Displacement*** and smoothes out flaws (actually more than flaws, it would be correct to define them as intrinsic properties of a raster image) which generate the stair-effect.
- At high resolutions, thanks to filters, for both 8 bit and 16 bit versions, the result is almost the same. This is not true for low resolution maps, which suffer the diversity of color-depth much more.
- Rendering times are shorter with filtering. This is because *VRay* does not have to compute GI on small "steps" of the unfiltered ***Displacement***. Also ***antialiasing*** reaps a benefit, as its calculation is faster thanks to filtering.

There are then also other factors which influence bitmap-made ***Displacement*** quality, namely the Pyramidal and Summed Area parameters.



■ **Figure 4.286**
Different results due to the use of Pyramidal and Summed Area filtering and the ***Filter blur*** parameter.

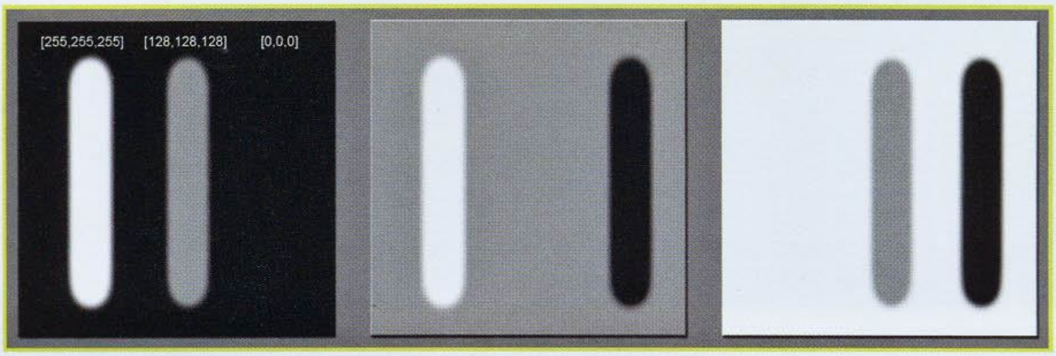
Filter blur - it defines the amount of Blur applied to the map. Analyzing the previous image it is clear that the map ***Blur*** influences ***Displacement*** significantly. High values smooth out ***Displacement***, at the cost of a loss of detail. One must pay attention especially to the Blur offset option, a parameter which has been described a few paragraphs ago.



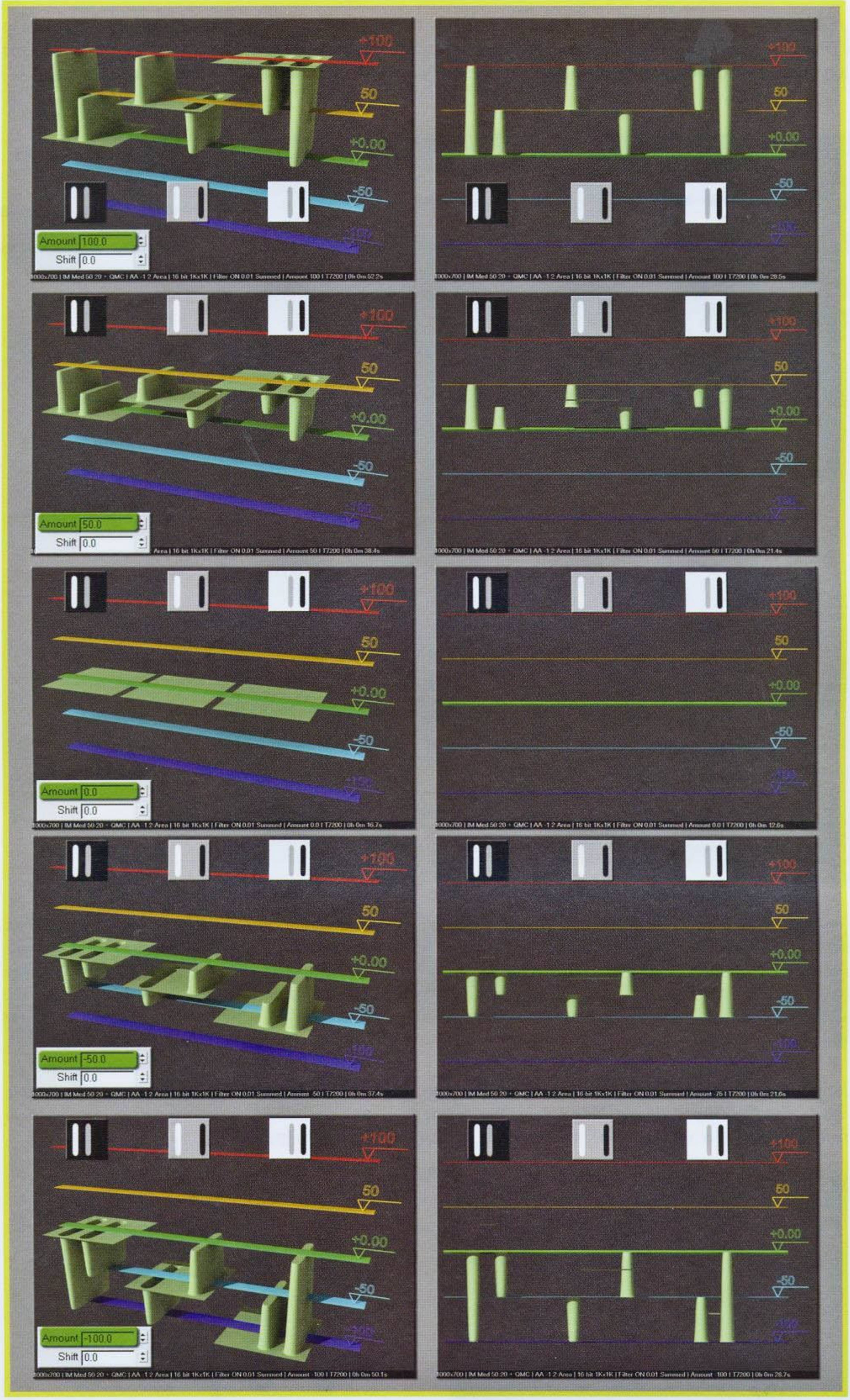
■ **Figure 4.287**
Influence of the ***Filter blur*** and Blur offset parameter in ***Displacement*** use. The ***Blur*** parameter, not to be confused with the Blur Offset option in the Material Editor, does not have any effect at all on ***Displacement***.

The ***Filter blur*** and Blur offset parameters are almost the same thing. Their modification influences the ***Displacement*** map in an identical way. Also, if one of these parameters is reset while the other has a positive value, or if one inputs that same value on both, the result is unaltered. Basically, ***Filter blur*** is no more than a handy short-cut which *VRay* programmers have placed for modifying the Blur offset parameter in the Material editor through the ***VRayDisplacementMod***. This way one can use the same map on different geometries, which can have different ***Filter blur*** settings, although sharing the same texture.

Amount - it defines the amount of ***Displacement*** applied to geometries. With value 0.0 no ***Displacement*** is applied. Values higher than 1 generate ***Displacement***. Negative values invert it.
In the next few tests three maps are used at 1000x1000 pixels resolution, 16 bit with grey tones.



■ **Figure 4.288**
The three maps used during examples.

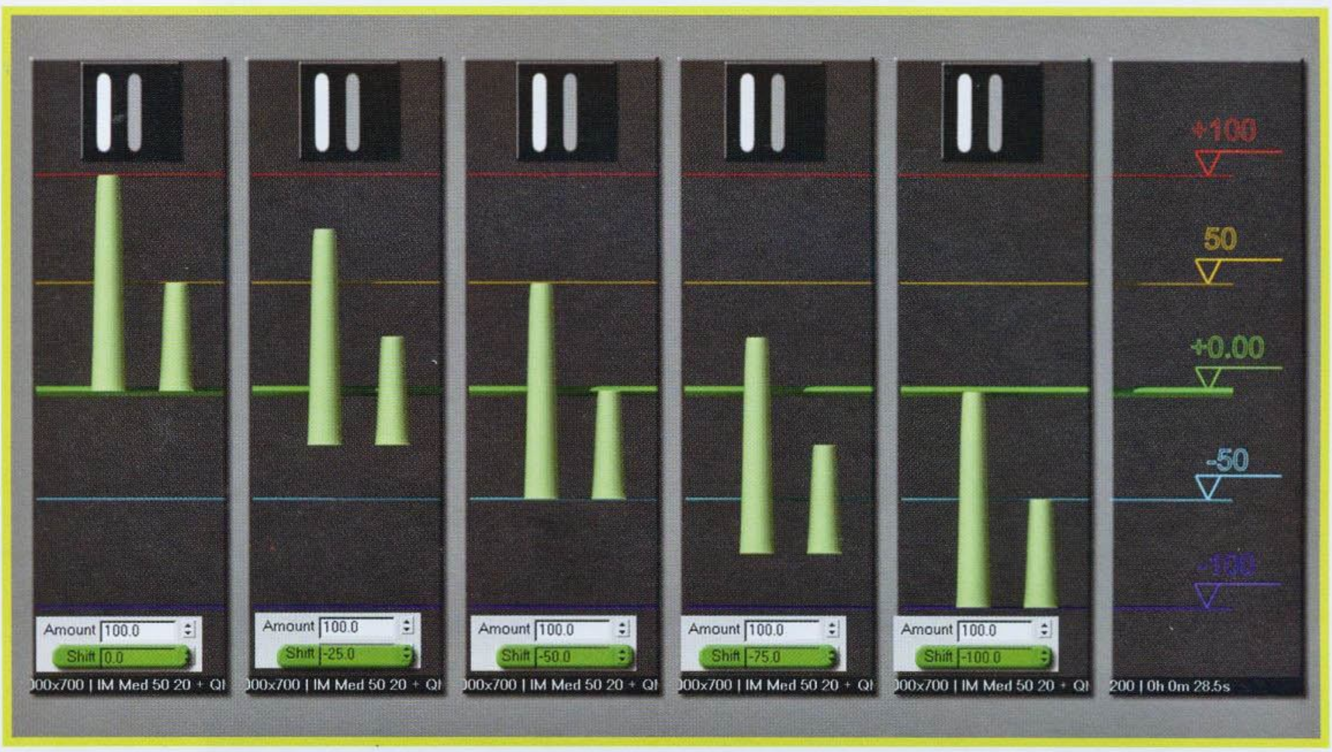


■ **Figure 4.289**
Use of various maps for explanation of the ***Amount*** parameter.

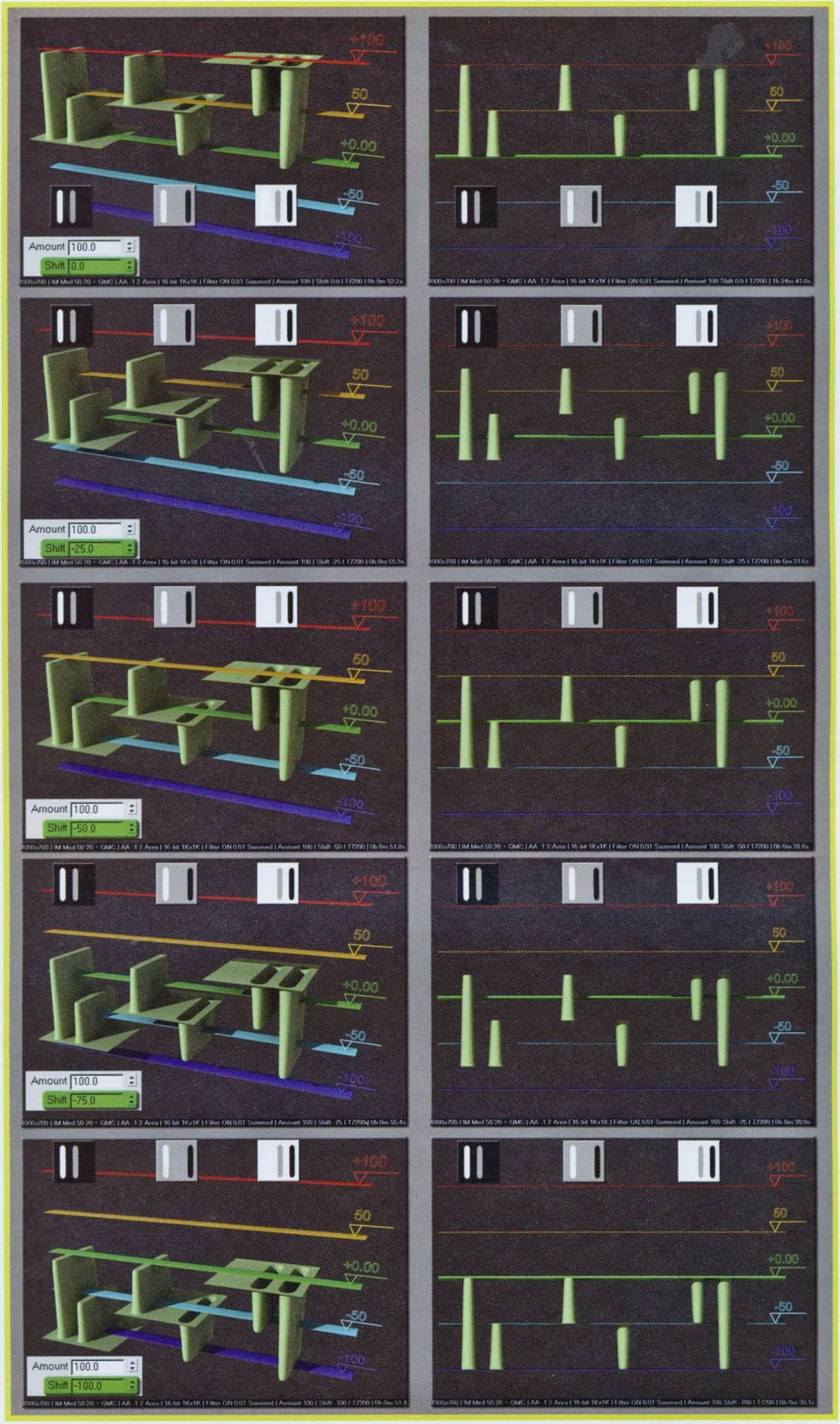
As one can see, the variation of the ***Amount*** parameter is of fundamental importance and generates completely different results for those parts of the model mapped with color textures which are not black. As the ***Amount*** parameter varies in fact, the black areas are not in the least bit influenced by ***Displacement***. On the contrary, the lighter areas are the ones which change the most when the parameter is modified. If ***Amount*** = 100, white zones reach an extension of 100 units, grey zones reach 50, that is, half the ***Displacement*** setting.

■ **Figure 4.290**
Amount parameter in the ***VRayDisplacementMod*** modifier.

Shift - it represents a constant value, which is summed or subtracted to map values used for ***Displacement***. Basically it moves the surface along its normal, taking on both positive and negative values. Its effect is constant for any color. In practical terms, ***Shift*** pushes ***Displacement*** as a whole block, upwards or downwards.



■ **Figure 4.291**
Shift parameter to be found in the ***VRayDisplacementMod***.



■ **Figure 4.292**
Different maps being used as examples for the ***Shift*** parameter.

Water level - by activating this parameter, which is by default deactivated, *VRay* "erases" geometry features in the area where ***Displacement*** is below the threshold specified by the ***Water level*** value.

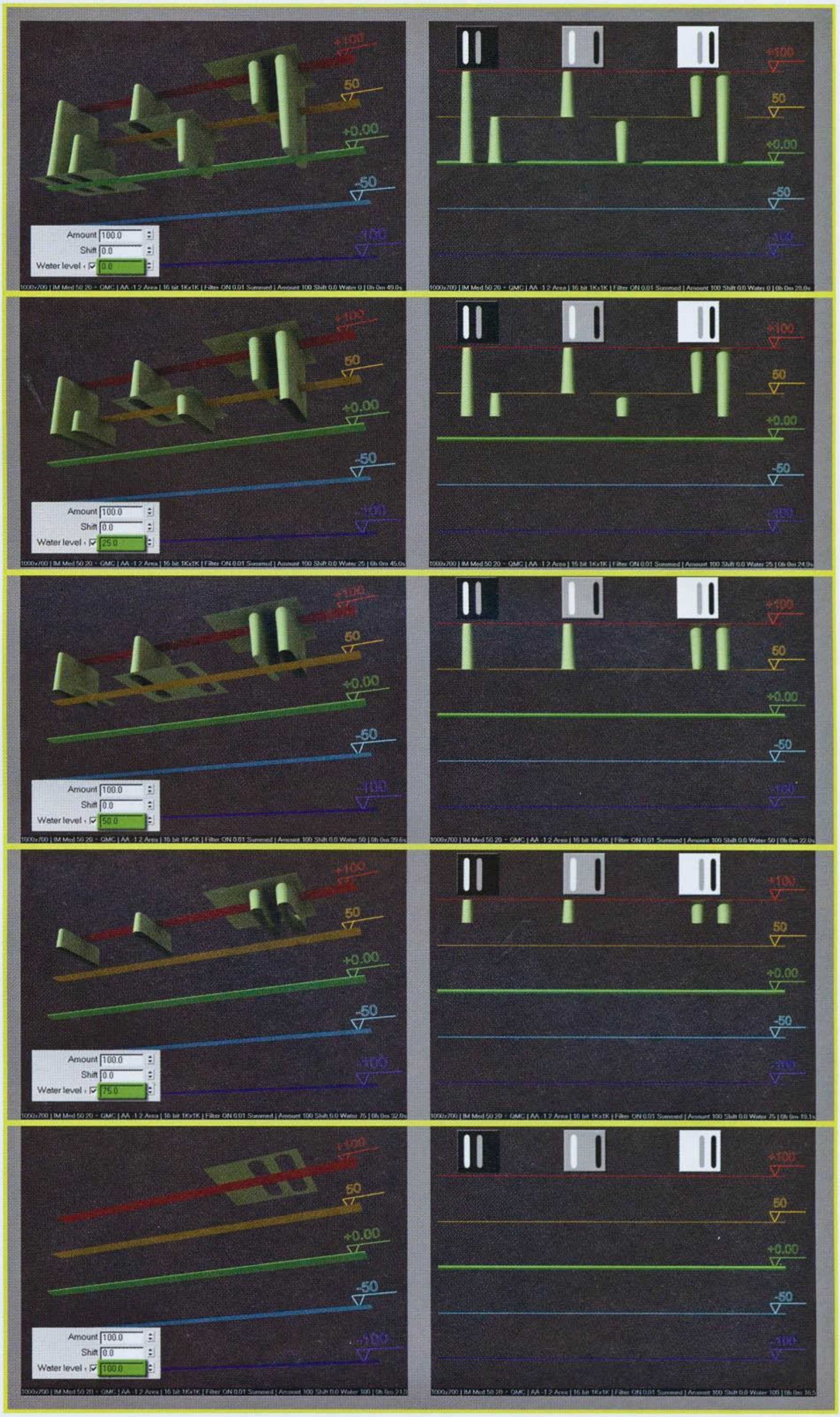
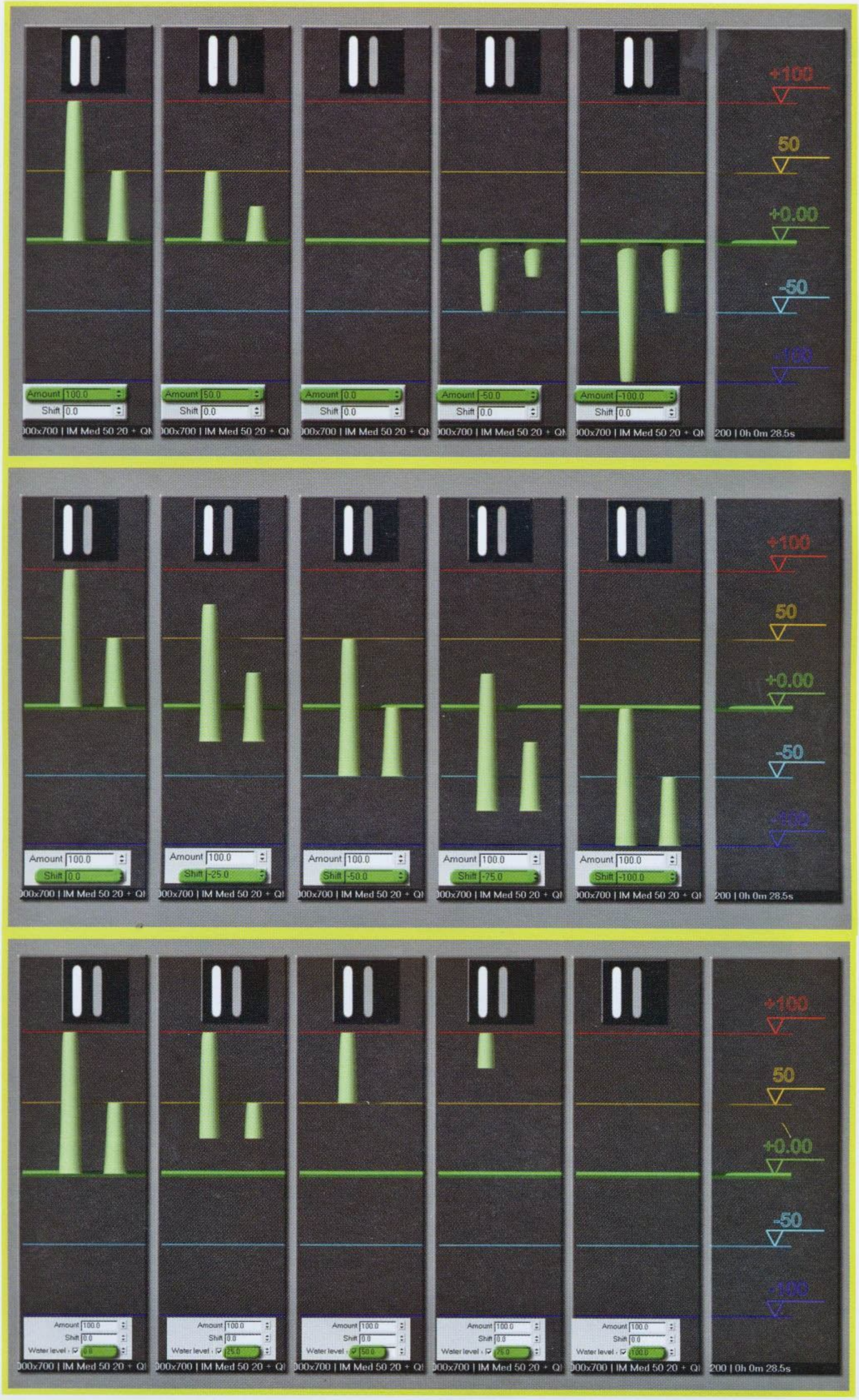
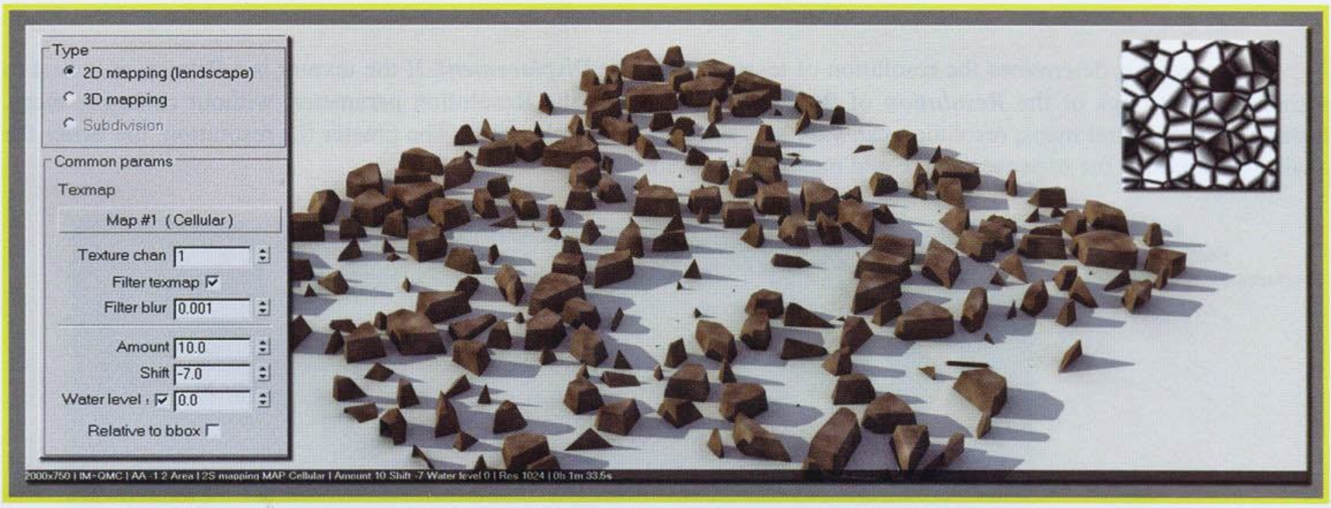


Figure 4.293
Water level parameter, in the ***VRayDisplacementMod*** modifier.



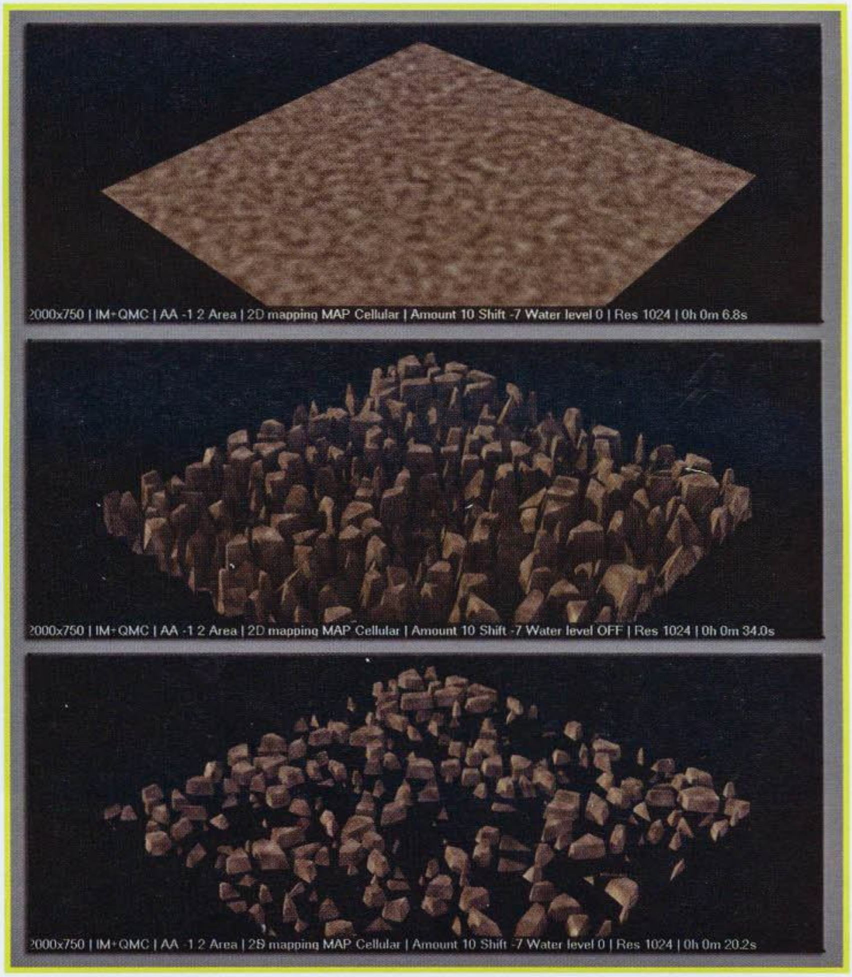
■ **Figure 4.294**
The three parameters, ***Amount***, ***Shift*** and ***Water level*** in action.

From the previous examples, one can observe how the ***Water level*** parameter nullifies the effects of ***Displacement*** under a certain threshold, beyond which geometry generation is blocked.



■ **Figure 4.295**
Summary example concerning the use of ***Amount, Shift*** and ***Water level*** parameters.

In this example, a procedural 3D map has been used in the creation of small rock fragments. The terrain, represented by the white surface plane has a height of +0.0, and so does the square 100x100 plane. ***Displacement*** has been applied via *VRay* with the use of a Cellular map. Obviously the result did not represent small rock fragments therefore ***Water level*** was applied in order to eliminate the unnecessary parts. As the ***Displacement*** plane is at a height of 0.0 and amount is equivalent to 10, The ***Shift*** parameter has been used to move the whole ***Displacement*** down by 7 units. This way, with ***Water level*** set to 7, the rocks are now 3 units high, and are perfectly adherent to the ground.



■ **Figure 4.296**
The three main phases for the creation of gravel via use of ***Displacement***.

Relative to bbox - see ***VRay: Default displacement***.

2D mapping

Resolution - this determines the resolution of textures used for ***Displacement***. If the texture is a Bitmap, it is best to maintain the settings of the ***Resolution*** of the raster map and of the Resolution parameter, without changing them. Instead for procedural maps, resolution can vary according to the user's needs. The greater the resolution, the better the quality becomes, at the expense of time and memory requirements.

■ **Figure 4.297**
Rendering at different ***Displacement*** resolutions via ***2D mapping***.

The ***Resolution*** option, as well as ***Precision, Shift by average*** and ***Tight bounds***, is available only when the ***2D mapping*** method is selected. The image is, in this case a black and white Bitmap with a resolution of 1024x1024 pixels. Which is the reason which lead to use of the highest resolution.

Precision - this parameter handles the curvature of the surface created by ***Displacement***. For flat surfaces this value can remain very low. In the case of completely flat geometries, this value should be set to 1.0. Curved surfaces instead require more precision. If this parameter is not high enough, black stains can appear on the ***Displace***. Low values sped up rendering time.

■ **Figure 4.298**
Demonstration of ***Precision*** parameter set to 1, 4, 8 and 16.

It can be seen how this parameter has no effect on flat surfaces. This is not true for curved models, where the ***Precision*** parameter allows a better result in the presence of GI. Increasing this value, the black stains tend to disappear and vanish completely when the parameter is set to 16. Rendering time increases but within acceptable limits. The default value of 8 is therefore correct for good quality renders.

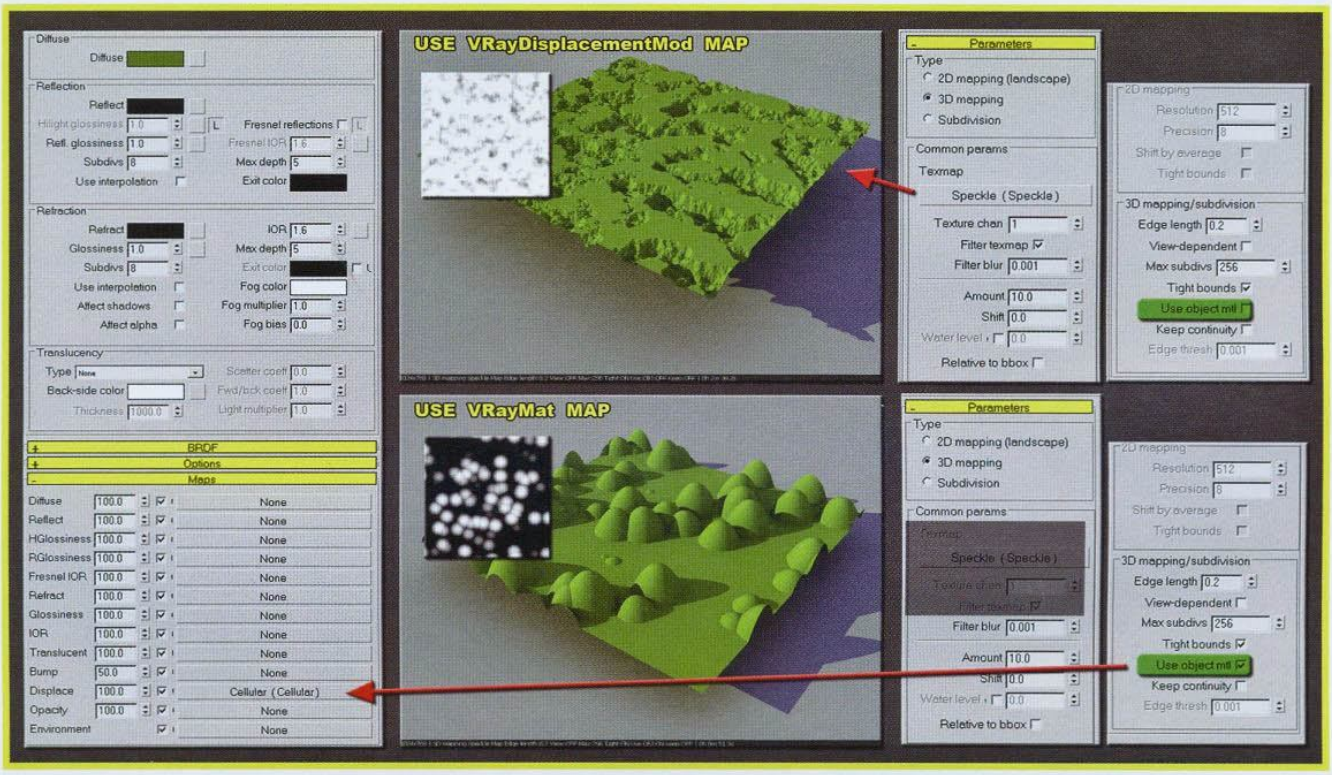
Shift by average - this parameter automatically moves ***Displacement*** using the value of the map average used for ***Displace*** (parameter which has been eliminated in ***VRay 1.5 FINAL***).

Tight bounds - this parameter permits one to calculate the room taken up by microtriangles in a more precise manner. In order to do this, a pre-calculation session is carried out during the rendering phase, which allows one to optimize ***Displacement***. The result is a faster render.

3D mapping/subdivs

Most of the parameters contained in this calculation mode are identical to those found in the ***VRay: Default displacement*** and can be seen in the relative chapter. There are however some new features, such as ***Use object mtl, Keep continuity*** and ***Edge thresh***.

Use object mtl - with this option active, the ***Displacement*** map is changed and the map in the Material Editor is used instead of the one in the ***VRayDisplacementMod***.

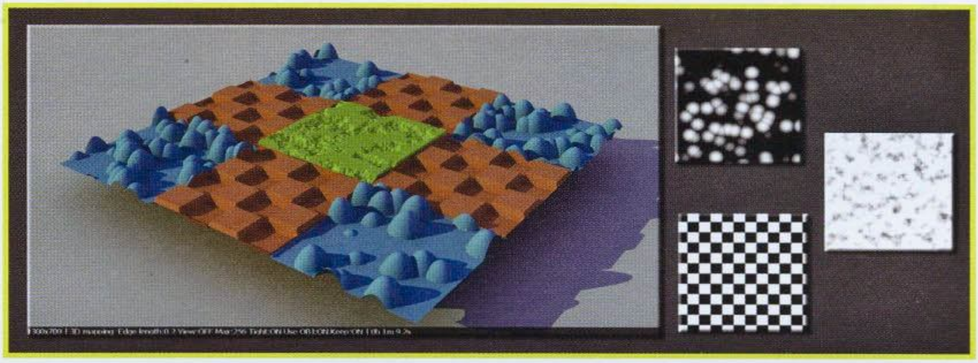


■ **Figure 4.299**
Practical example of the ***Use object mtl*** parameter.

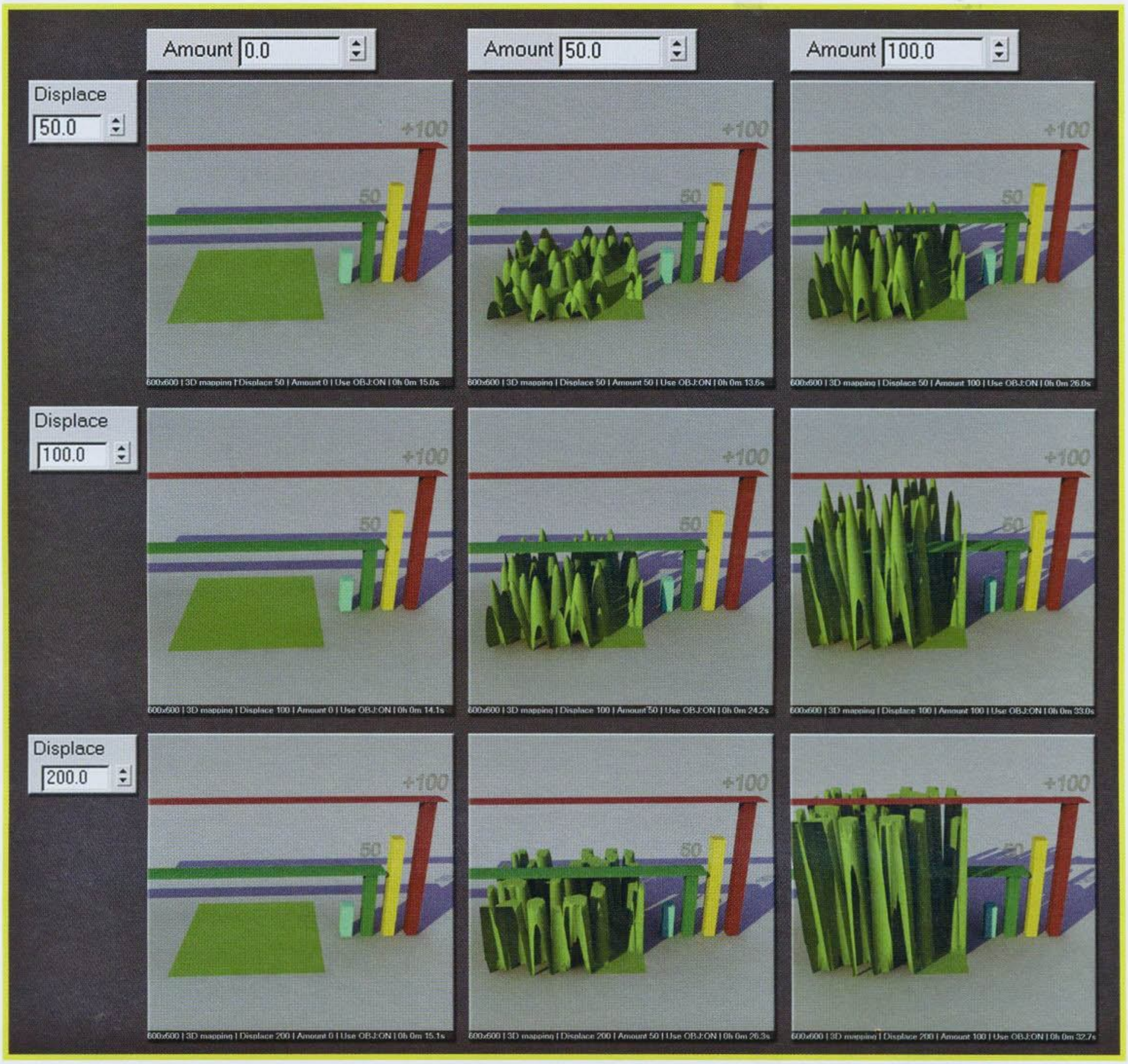
In the example we have just shown, a ***VRayDisplacementMod*** is applied to a surface. A 3D Speckle procedural texture has been used for ***Displacement***. By using procedural 3D maps with the 3D mapping method, map coordinates can be ignored. ***Displacement*** is in any case correct and all modifier parameters are active. After, the surface was assigned to a ***VRay*** shader, its ***Displace*** channel being occupied by a Cellular map. For the moment, this texture does not influence ***Displacement*** at all, because this is handled by the modifier. But if ***Use object mtl*** is activated, the map in the modifier appears as opaque and cannot be changed, as ***Displacement*** will use the map inserted into the shader channel.

This parameter can be extremely useful if one wishes to apply more than one ***Displacement*** to the same geometrical objects: it is in fact enough to apply multiple shader IDs with different ***Displacement*** maps, apply the ***VRayDisplacementMod*** and activate the ***Use object mtl*** option. One must bear in mind that all the parameters in the ***VRay: Default displacement*** rollout do not influence ***Displacement*** obtained via use of modifiers at all.

■ **Figure 4.300**
Geometric model mapped with more than one texture in order to obtain a multiple ***Displacement*** through activation of the ***Use object mtl*** option and a Multi/Sub-object shader.

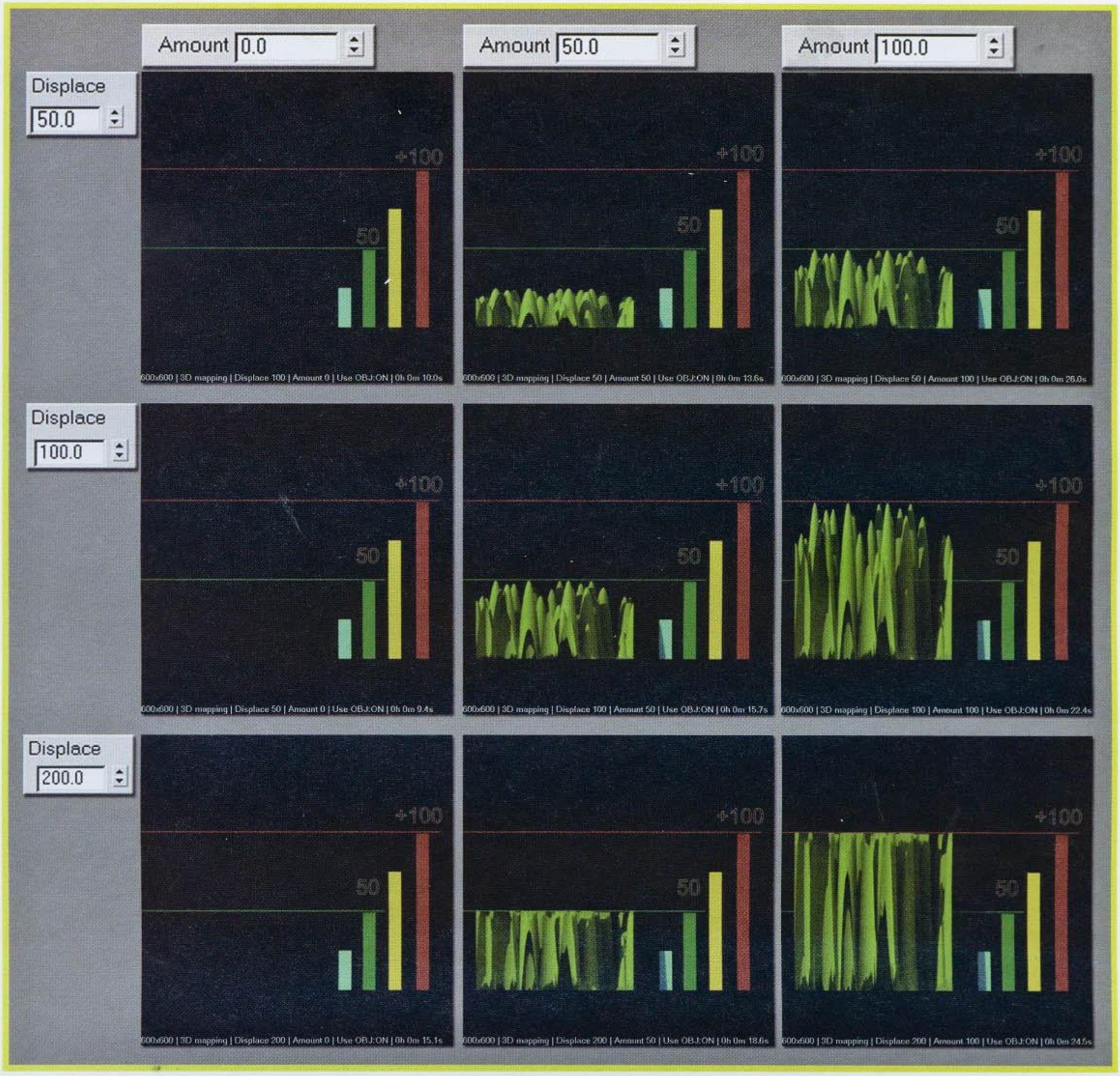


At this point we must introduce a new element: the ***Displace*** map multiplier of the ***VRayMtl*** shader.



■ **Figure 4.301**
Use of the parameters ***Displacement*** (***VRayMtl***) and ***Amount*** (***VRayDisplacementMod***) for generating ***Displacement***.

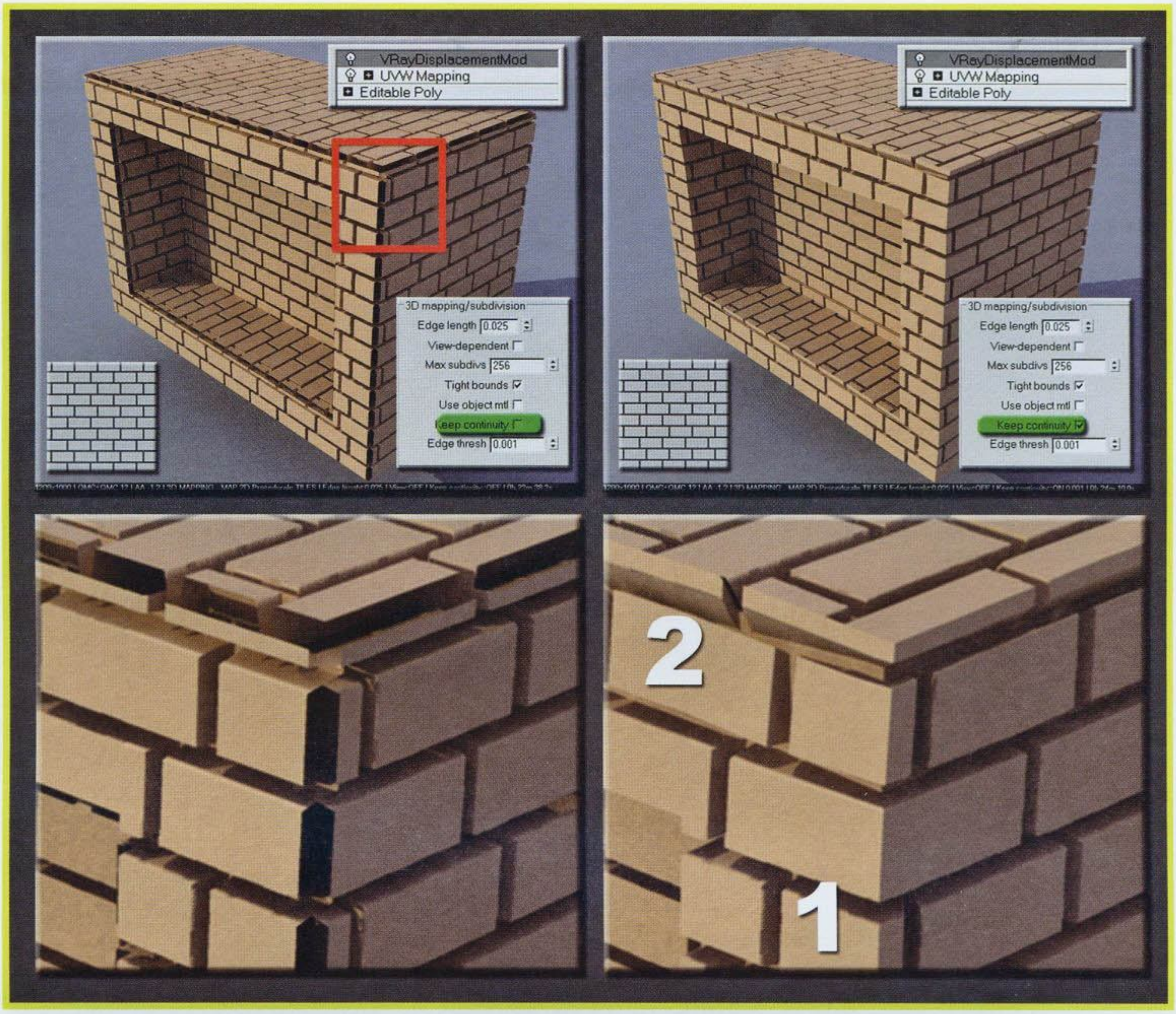
The correct solution, whereby an ***Amount*** value of 100 effectively corresponds to a ***Displacement*** of 100 is that by which ***Displace*** value for the shader is equivalent to 100. Instead using a constant ***Amount*** value, but varying the amount of ***Displace*** (for instance 50), one obtains a ***Displace*** reduced by 50%. At ***Amount*** = 100 an actual ***Displacement*** of 50 units is obtained.
A strange effect is caused by ***Displace*** values above 100. With ***Amount*** set to 100 units and ***Displace*** set to 200, the rendered ***Displacement*** is 100 units, but it is truncated at the top. This is because 3D mapping fulfils ***Displacement*** between 1 and 0 [0,255], between black and white. Values above or below this are clipped, cut off. The same thing happens with ***Amount*** = 50 and ***Displace*** still at 100. Here the rendered ***Displacement*** is 50, but it has still undergone clipping.



■ **Figure 4.302**
Front view of the previous test.

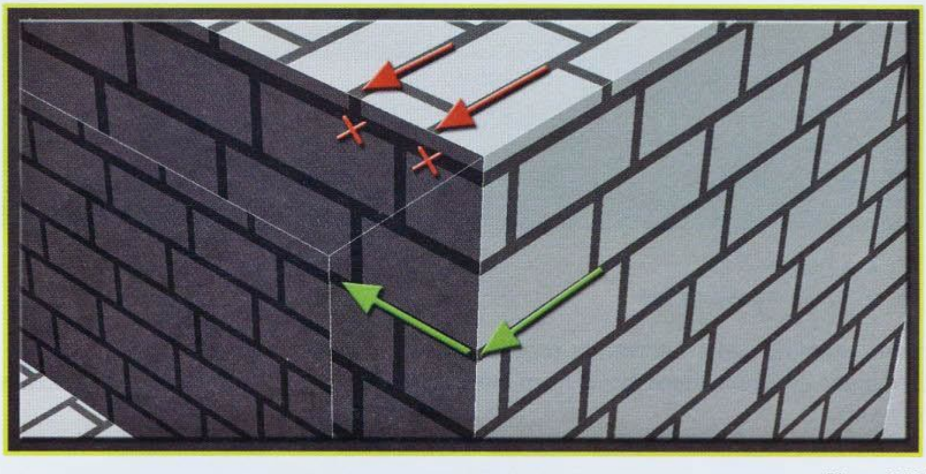
Keep continuity - with this parameter ***VRay*** attempts to maintain continuity as much as possible between surfaces with different smoothening qualities or different shader IDs. Using shader IDs is not an advisable method for continuity in surfaces subject to ***Displacement***. Instead it is preferable, for this purpose, Blend type maps or Vertex Color in order to blend different ***Displacement*** maps.

Edge thresh - when the ***Keep continuity*** parameter is active, ***Edge thresh*** becomes available. It is therefore possible to control the amount of blending between maps applied to various surfaces with different shader IDs. ***VRay*** only assures the continuity of the sides but not of corners. This means that there can be possible interruptions, instead for sides no problems arise. Thus it is best to keep the ***Edge thresh*** as low as possible.



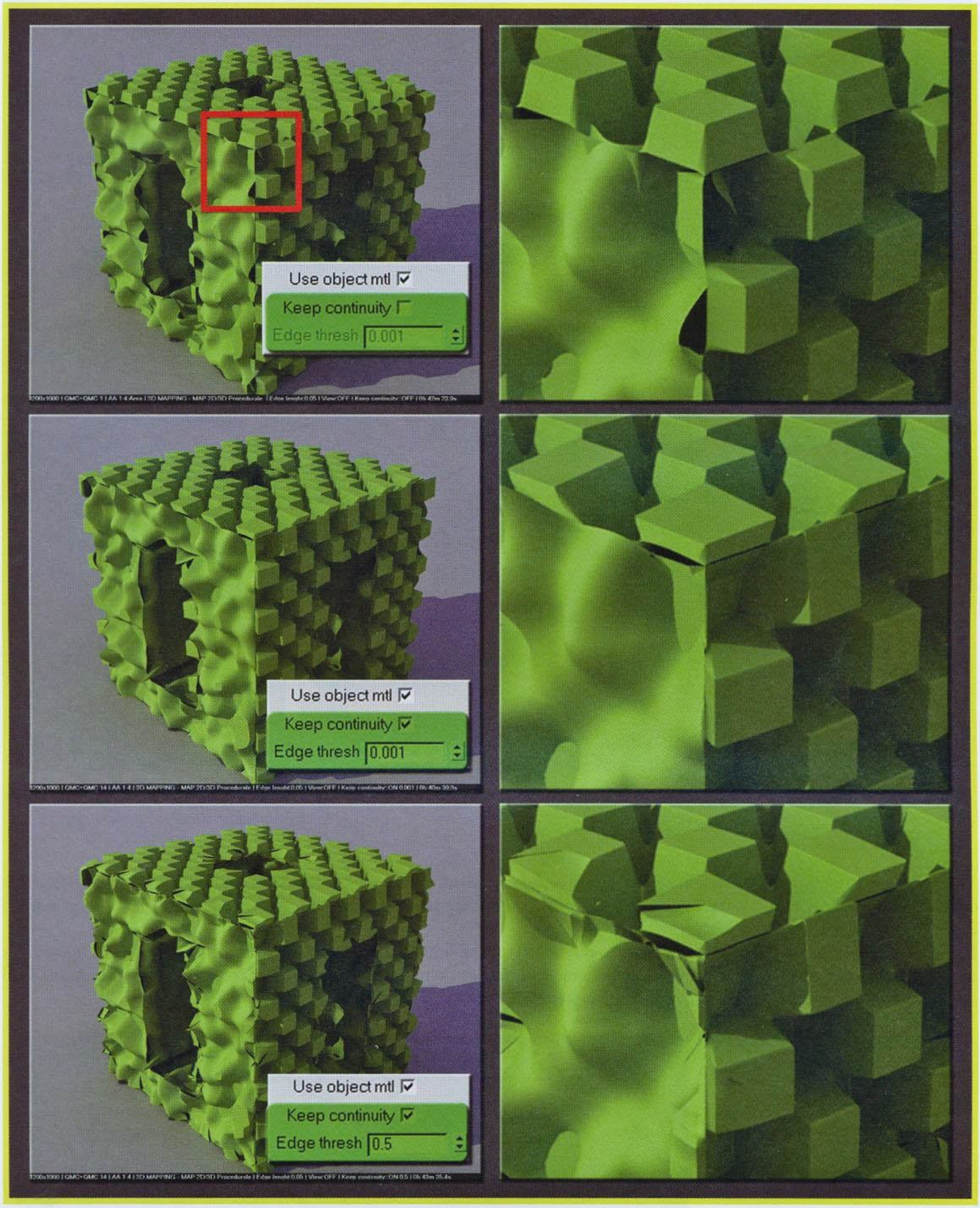
■ **Figure 4.303**
Practical application of the ***Keep continuity*** parameter.

The effect of using the ***Keep continuity*** parameter is quite evident. Thanks to its use, the open sides of the mesh become continuous, giving a realistic feeling of ***Displace*** in three dimensions. There are, however, some flaws which ***Keep continuity*** cannot in any way correct.
In the spot area indicated by the green arrows, continuity of the bricks is guaranteed by the fact that the 2d map itself is continuous. In the areas indicated by red arrows instead, for obvious 2d texture mapping reasons, continuity is lost and we can witness a strange border where the two areas unite.



■ **Figure 4.304**
Continuity and lack of continuity in this 2D map.

Other examples for ***Edge threshold*** parameter.

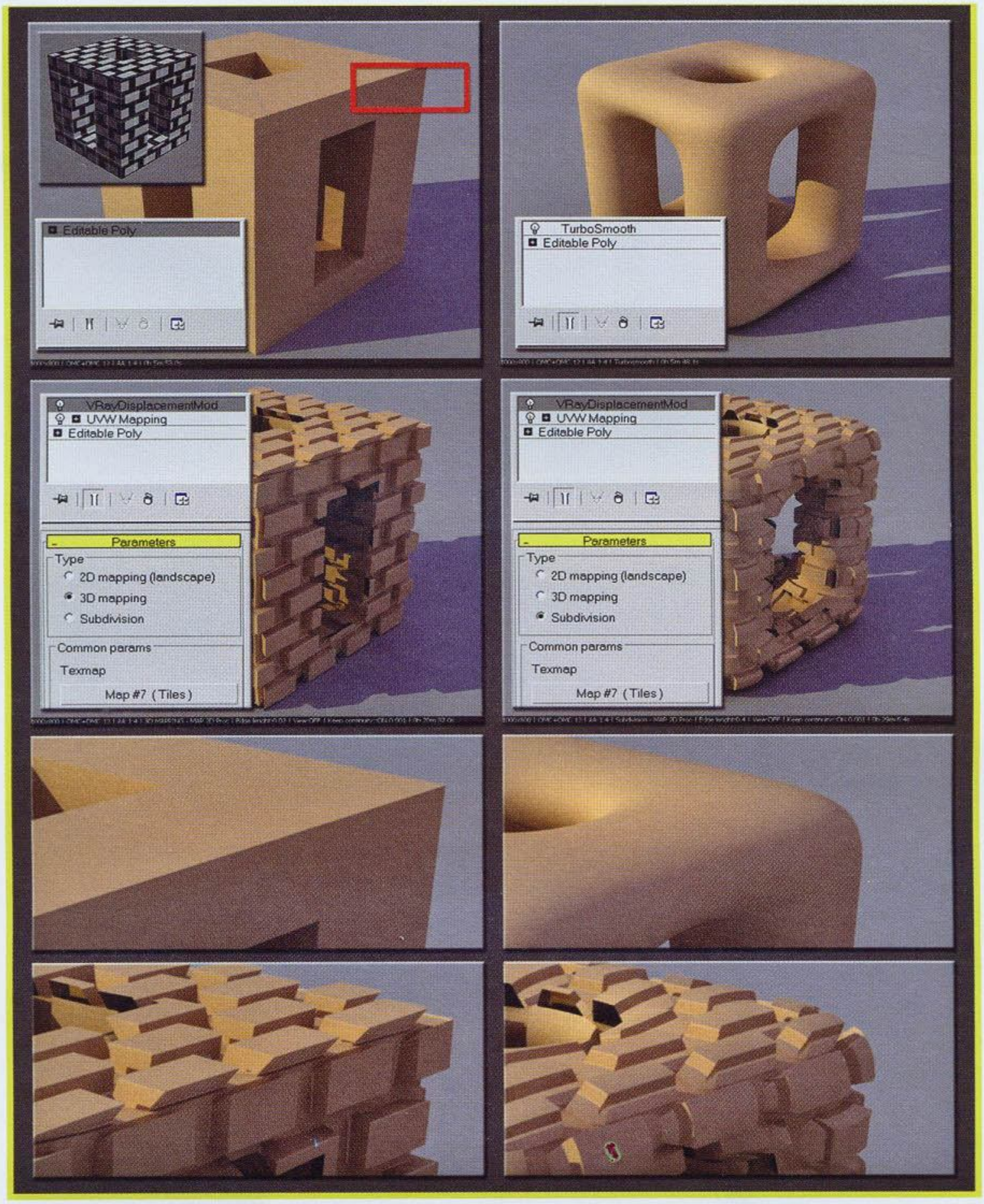


■ **Figure 4.305**
Use of the ***Edge thresh*** parameter.

In this example some interesting results can be seen. The first one regards the use of the ***Keep continuity*** parameter. Extremely different maps have been used in order to observe how *VRay* attempts to maintain continuity between polygons.

There is too great a difference between the Noise and Checker maps for one to possibly expect a perfect uniformity along edges and even the union between these maps in general is difficult. An interesting attempt is to make the Checker texture continuous by tilting the edges of the box. Also the ***Noise*** map improves the result with the use of the ***Keep continuity*** option, but there are still problems. Instead the continuity between polygons mapped with the Noise texture and those mapped with the Checker texture are excellent. The border is quite linear and artefacts are scarcely visible.
This is were the ***Edge threshold*** parameter comes into play. This option attempts to maintain continuity between surfaces which have different shader IDs. Obviously these two maps, which have been used on purpose in order to weigh upon ***VRay***, are much too different to each other. The attempt of ***VRay*** to blend polygons along the axis with shared common edges is clearly visible. One can see how the Noise map tries to enter the Checker map territory, breaking up its uniformity. Also the Checker map tries to place its polygons within the Noise map.

The explanation of the last mode, the ***Subdivision*** system, is very simple. It is basically a ***Displacement***-based application of the Meshtooth or Turbosmooth modifiers. In practice, during rendering, the geometry is both involved and smoothed by ***Displacement***, all in one go.



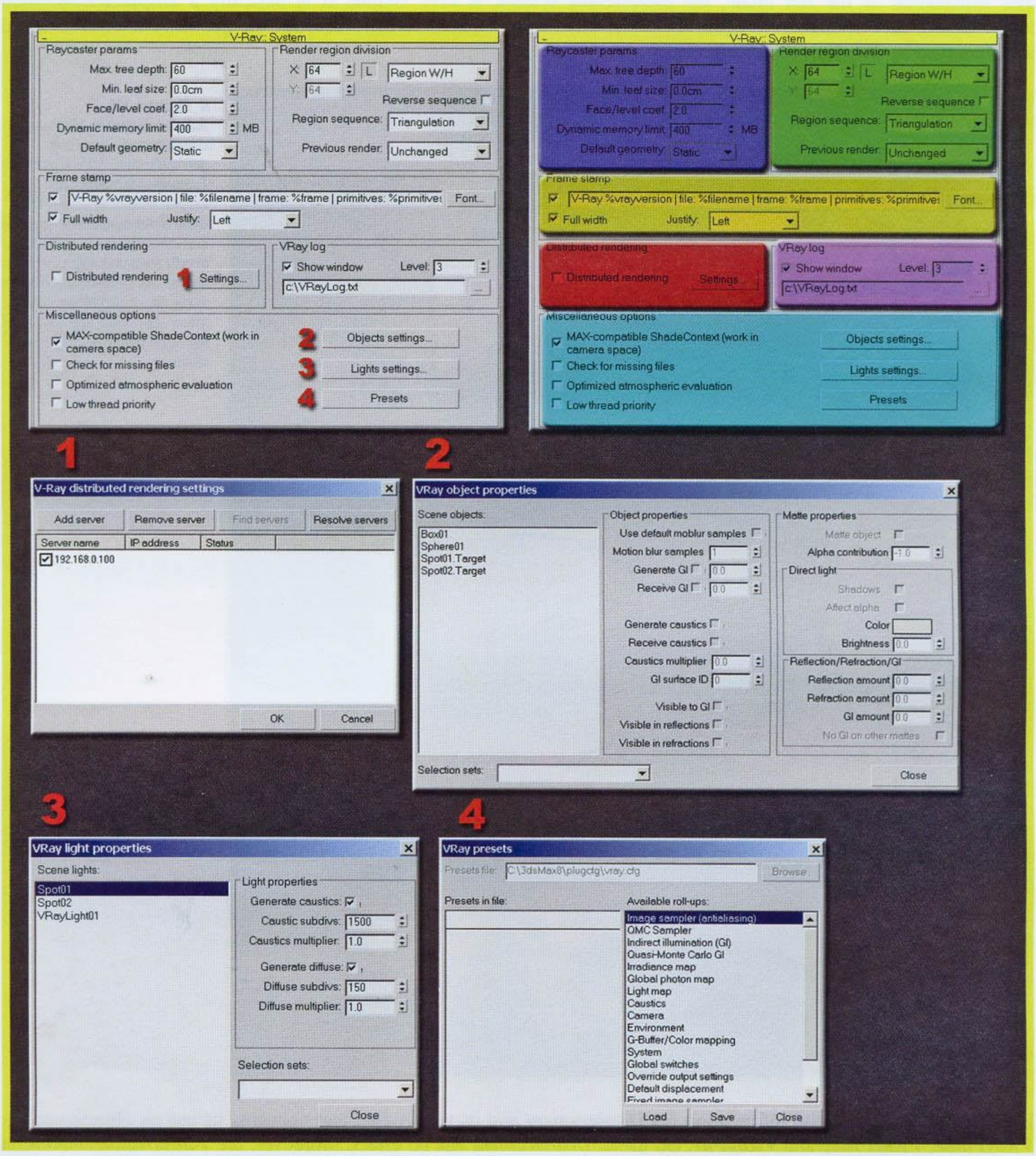
■ **Figure 4.306**
Subdivision mode allows one to simultaneously apply both geometric subdivision typical of the Turbosmooth modifier and ***Displacement***. By setting the value of ***Amount*** to zero one can even smooth out the geometry in exactly the same way as when one applies the Turbosmooth modifier alone. Rendering times however are nowhere near and are higher in the ***Subdivision*** displacement mode.

Lastly, there is some useful information for ***Displacement*** and its use. A first part concerns its use on objects such as ***VRayPlane***, ***VRayProxy*** and ***VRayFur***. On these, ***Displacement*** has no effect. Moreover, the ***VRayDisplacementMod*** does not take into account the selection of surfaces. In fact the modifier applies ***Displacement*** on the whole object. If one wishes to apply ***Displacement*** on a specific area of the geometry, it is necessary to color the map of black in those areas where this effect is not wanted.

VRay: System

INTRODUCTION

The ***VRay: System*** rollout can be defined as a group of parameters of various types, difficult to classify as we have so far.



■ **Figure 4.307**
The ***VRay: System*** rollout and all its options, including the extra windows.

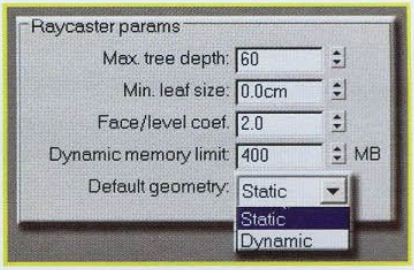
It is subdivided into 6 distinct areas, each one containing parameters which differ greatly from one another. One can also access, via a few commands, 4 other sub-sections which concern the settings for network rendering, light property and shader settings and rendering presets.

The six main areas instead are:

- ***Raycaster parameters***. This series of parameters grants *VRay* management and use of system memory.
- ***Render region division***. It is possible to specify the size of rendering buckets, calculation order and other ***VFB*** options.
- ***Frame stamp***. It is a simplified version of the editor in ***VFB***. It allows one to add a caption to the render with which to input information such as *VRay* version, rendering time and so on.
- ***Distribuited rendering***. One can manage network rendering by setting clients and activating or deactivating them.
- ***VRay log***. This allows one to control, activate and modify the settings message window in *VRay*.
- ***Miscellaneous options***. There are some parameters with varied purposes. The most important ones are those which can be activated via three buttons, each of which leads to a new window.

PARAMETERS

Raycaster params



■ **Figure 4.308**
The ***Raycaster*** parameters in the ***VRay: System*** rollout.

This series of commands, which one might define advanced, allow one to modify RAM management in *VRay*. It can happen at times that *VRay*, and consequently also 3ds Max, freezes due to an excessive memory demand. In order to avoid this it can be useful to change the ***Raycaster*** parameters. In general, by reducing the amount of RAM assigned to *VRay*, rendering time goes up. And vice versa, with more RAM available (for Windows XP 2 GB are recommended. In a 64 bit system such as Windows XP 64 or Vista 64 bit, *VRay* can exploit a greater amount), renders can benefit, making the calculation faster. Before facing the parameters in this section it is necessary to understand a few precise terms, like ***BSP tree***.

Binary **S**pace **P**artition trees, or ***BSP trees***, is a piece of terminology which is extremely important for anyone working in the field of 2D or 3D. It was introduced in 1980 by Fuchs, Kedem and Naylor, who wrote two documents, "Predetermining Visibility Priority in 3-D Scenes" and "On Visible Surface Generation by A Priori Tree Structures" which describe the use of ***BSP trees*** and their implementation in the field of CG. At the beginning the three researchers focused their efforts on improving the quality of renders in static scenes. Later on, other researchers exploited this algorithm for generating shadows and manipulating dynamic scenes. In fact the hardware of that period was not as sophisticated and powerful as it is today; tools like the Z-buffer had not yet been invented. There was therefore the need to speed up polygon calculation in videogames.

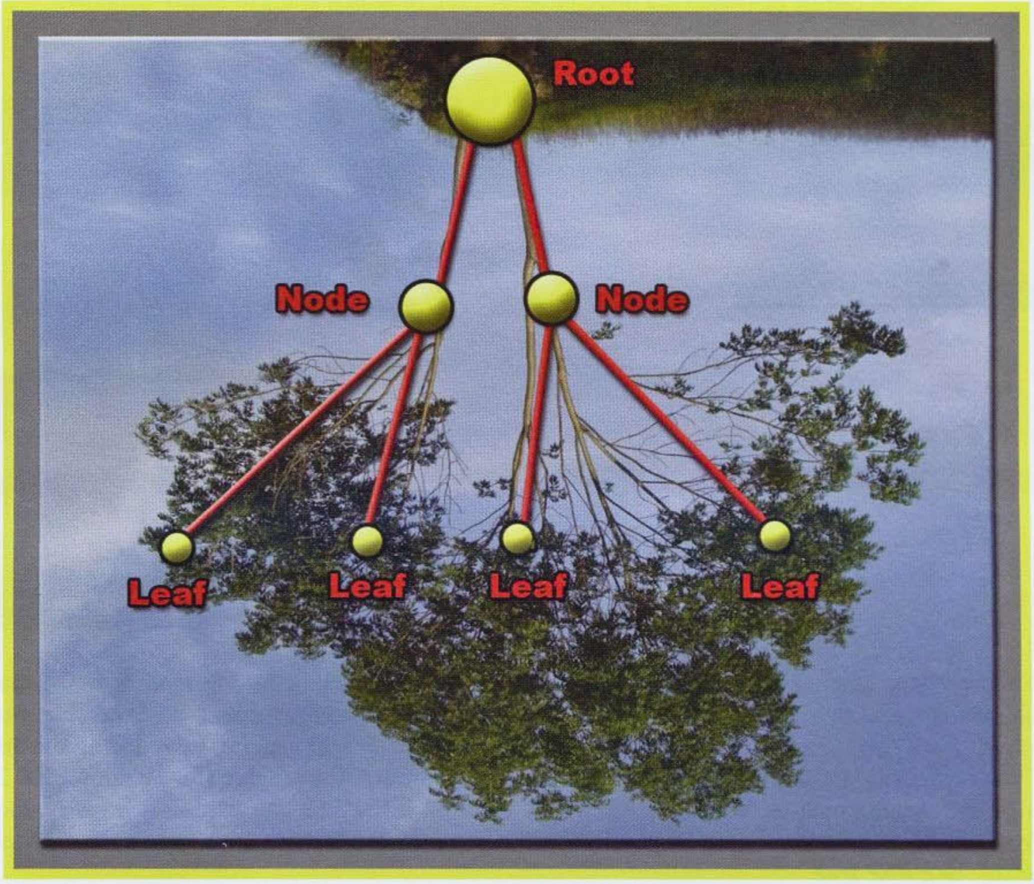
The first real use of ***BSP trees*** is later in 1993, when the game Doom was developed, a legendary development by John Carmack and John Romero. After its success, many other software houses understood the importance of this algorithm and a game series was born, inspired by the game by ID software, amongst which also Quake.

But what exactly is the Binary Space Partition tree? The best way to explain it is to understand each single word which makes it up.

- BINARY: This means "two parts"
- SPACE : This simply stands for the area one is working in. In the games world, by space one means all the rooms, walls and locations which make up the level. In the field of CG it can mean the space occupied by the geometrical objects in a scene.
- PARTITION: this means "to divide". As in dividing a birthday cake into parts.
- TREE: in computing language, "tree" takes on a complex meaning and takes on the name of Binary Search Tree. These structures are used to implement indexes and search/order functions.

It is clear now that a "tree", in the world of computers, allows one to manipulate and manage information. Unlike what happens in nature, a digital tree is represented with the root at the top of the element and the leaves at the bottom. These are basically just nodes in a structure which makes up the data in the scene.

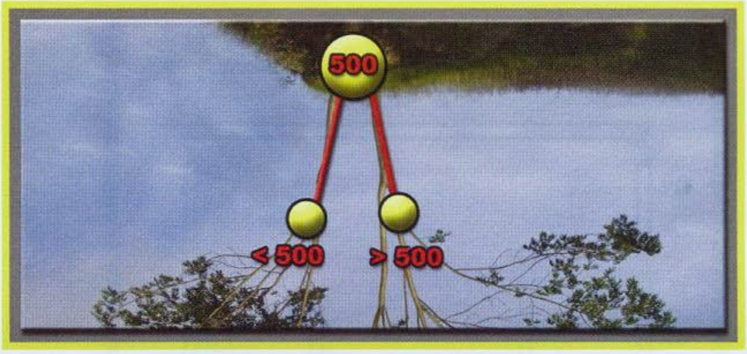
Figure 4.309
Tree structure.



Amongst these nodes there are relations which can be defined as parent/child type. Usually the nodes in these structures can have both "relatives", being both a child and parent at the same time. Obviously the root can only be a parent being at the top of the whole structure, whereas the leaves can only be "offsprings".

For what reason is it preferable to describe an array by means of a tree structure as opposed to a simple list of elements?

Imagine a list of 1000 objects in a sequence. If we wanted to look up number 999 we would have to scroll through 999 objects before reaching it. Now instead of a list, imagine constructing a tree structure, were a root branches into two nodes: a binary tree. Starting from the root we can place the first 500 numbers on the left and the other 500 on the right. This way we have halved the number of searches to be carried out in order to find number 999, which we know in advance to be in the right-hand group.



■ **Figure 4.310**
Example of a binary tree structure system.

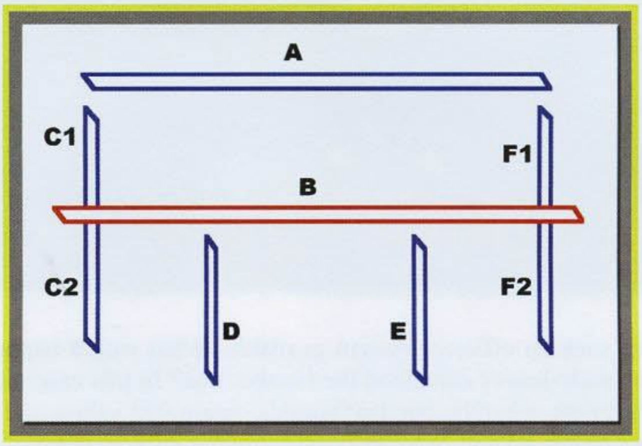
Obviously one does not always have such an efficient system available. What would happen for example if the root contained the number two, one of the node-leaves contained the number one? In this case there would be an enormous disproportion in the distribution of numbers, whereby one leaf would contain 998 values and the other one only one. In actual fact there are certain measures one can take in order to avoid problems such as these.
So we have now defined the ***BSP tree***, that is a binary structure which thanks to particular selection functions, maintains the system balanced by determining specific values in each node. Its aim is to divide a 2D or 3D space into parts which can be quickly looked up and placed into a list, from which one can obtain information about how these small portions of space are related to each other. One can for instance store data such as the distance between polygons, the position and so on. In practical terms, the ***BSP tree*** literally divides the space being analyzed. The process of subdivision proceeds until the parts which have been obtained are small enough for their aim.

Picture an imaginary plane flight boarded in order to visit four friends, one who lives close to your house, two in France but not in the same city, one in the USA and one in Australia. In order to travel the least possible, the first thing which must be done is to plan the journey. Otherwise one could risk going to France, then to Australia and then back to France to visit the second friend, and basically go out of one's way uselessly. One method to find an order by which to visit them without wasting time and money could be to divide the world map into two parts. On the right are the friends who live further away on the left the ones who are closer. Then one takes the two parts divided in the first selection and one divides once again. Now we have four portions. These must now be placed by order of distance. Now one keeps dividing the area into smaller parts until these are of a size which contains one's single friends' homes. At the end we will be left with a list of all the areas of the world, organized by order of distance between oneself and their homes. In this case we can ignore all those areas where none of them live.
Although this is an extremely simple example and does not explain accurately the way the ***BSP tree*** works, one can start to understand how this system can be particularly efficient in the field of management and organization, especially in extremely complex scenes.

One of the first large-scale commercial uses of this method was during the creation of the videogame Doom. During the game the computer must calculate and recreate everything present in the view. This happens for every instant of the game, unless one remains still. In order to avoid problems, lack of fluidity or lagging the PC must recreate the scene extremely fast, even 30 to 60 times per second. This requires both calculation power, which was not much at the time and an efficient organization of data. In a single game level there can be thousands of polygons. A ***BSP tree*** is none other than a list of polygons of the scene containing the necessary data for the game. For the computer to reproduce these geometries accurately, the game must quickly decide which polygons are to be viewed and in what order. Before these operations it is possible to know the whole layout of the level, because the game programmers have seen to this during the projecting and programming of the game. This information, such as the position of walls and the pavement, etc. does not change during the game. This allows one to know the list of polygons to be viewed in the scene before the game begins. As there can be tens of thousands of polygons, the list is very long and searching for the geometrical objects to be viewed in real-time can take a long time and during a real-time game there is no time to be wasted!

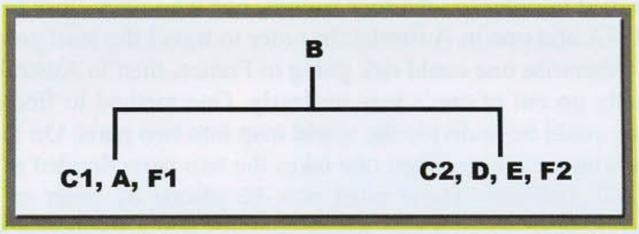
The solution was thus found in the ***BSP tree***. It can also be used for other purposes, such as removing the lines of hidden polygons, detecting collisions, managing polygon visibility, shadow calculation etc...

The ***BSP tree*** therefore allows one to reproduce or memorize the polygons in a scene. It is a well-known fact that a plane divides space into two semi-spaces: this is the concept which lies at the base of the creation of ***BSP trees***. With reference to the following image, having chosen a polygon, in this case B, one calculates the equation of the plane which passes through that polygon and then one classifies the other polygons compared to this plane. They can thus coincide, cut through plane B perpendicularly or be in front or behind it. The polygons which intersect the plane are divided into two other polygons so that one is then behind and the other in front of plane B: in this case C1, C2 and F1, F2.

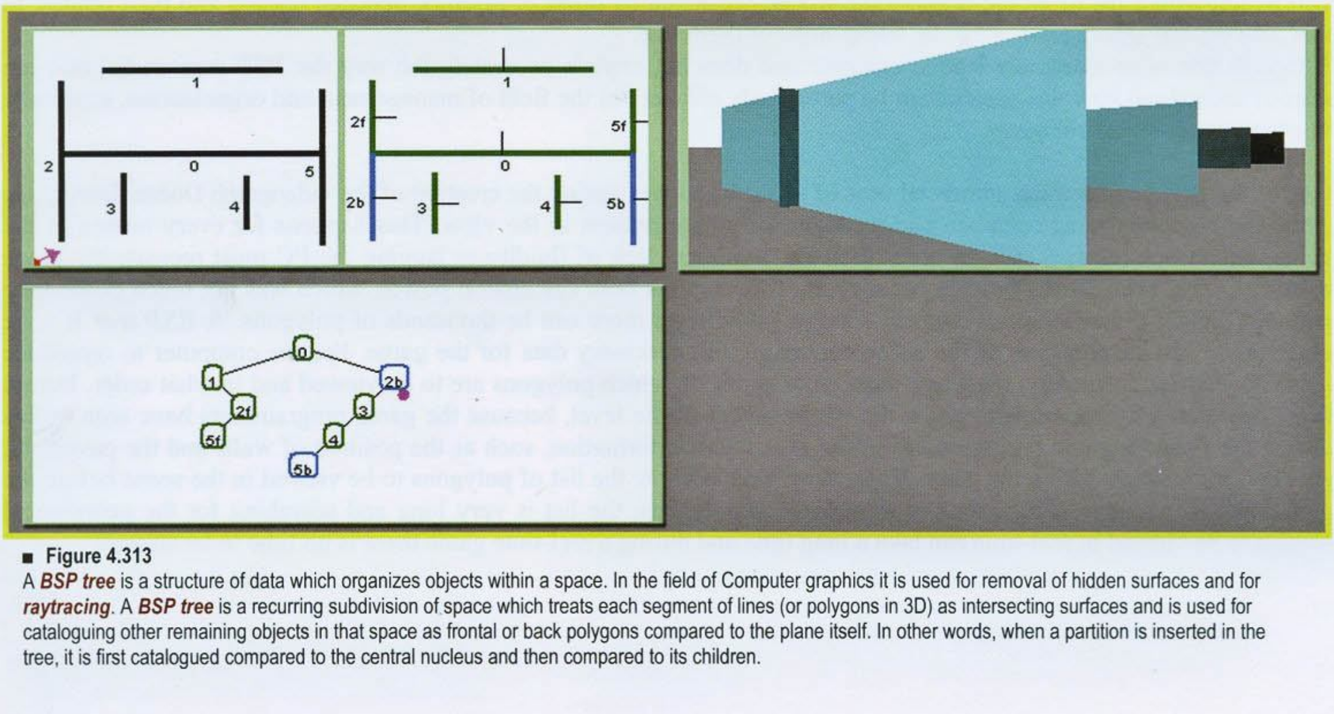


■ **Figure 4.311**
View from above of a hypothetical scene made up of six walls.

At this point one creates a binary tree which has polygon B for its root (chosen as the partitioner of the space), C1, A, F1 polygons as its left branch (behind the plane) and for its right branch the frontal polygons C2, D, E and F2. The algorithm is recurring: one chooses a new partitioning polygon between those contained in the branches and one applies the same classification technique again. This way one obtains a binary tree which represents spatial relations between polygons. By crossing the tree one can draw the scene starting from the closest polygon (front-to-back) or from the furthest away (back-to-front).



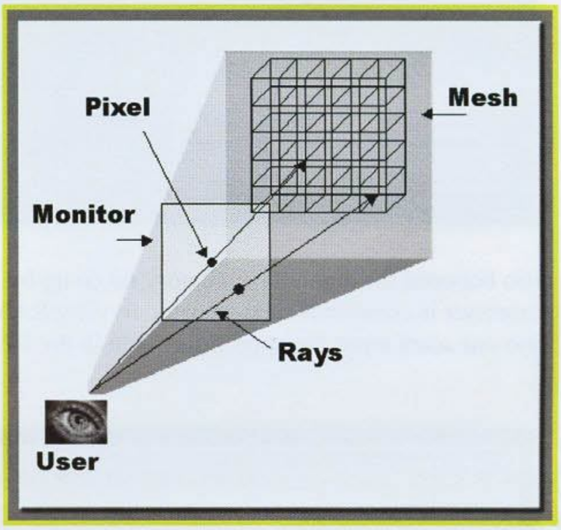
■ **Figure 4.312**
BSP tree created from the previous scheme.



■ **Figure 4.313**
A ***BSP tree*** is a structure of data which organizes objects within a space. In the field of Computer graphics it is used for removal of hidden surfaces and for ***raytracing***. A ***BSP tree*** is a recurring subdivision of space which treats each segment of lines (or polygons in 3D) as intersecting surfaces and is used for cataloguing other remaining objects in that space as frontal or back polygons compared to the plane itself. In other words, when a partition is inserted in the tree, it is first catalogued compared to the central nucleus and then compared to its children.

Now that we have understood the meaning of ***BSP tree***, we must face a new subject, ***RAYCASTING***.

Raycasting is not another name for ***Raytracing***, but can be defined as a system similar to ***Raytracing***, certainly a simplified version but faster. Both are used for rendering 3D scenes on a 2D system, the computer screen, by tracing the rays of light contained in the scene from the eye of the observer back to the light source. ***Raycasting***, unlike ***Raytracing***, does not calculate new tangents or the direction of hypothetical reflected or refracted rays. Because of this, ***Raycasting*** cannot calculate effects such as refraction or reflection. This "flaw" has in time revealed itself as an asset, allowing ***Raycasting*** to be much faster for certain types of calculations. This has led to the use of ***Raycasting*** in environments such as videogames in real-time. One of the first was Wolf3D, where a number of rays would be launched from the observation point of the scene. When these rays hit the object, the distance obtained would be used to decide what was to be shown on the screen.

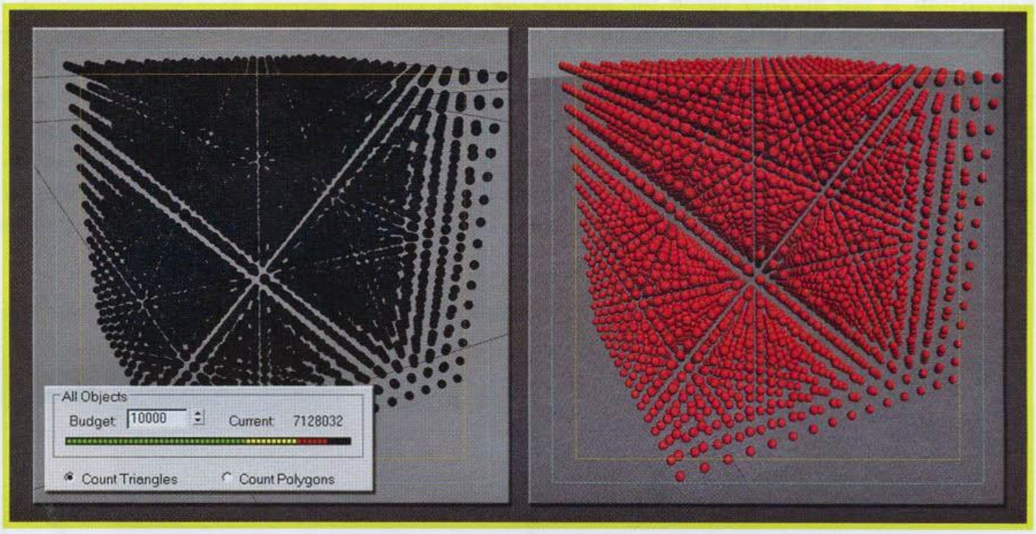


■ **Figure 4.314**
Raycasting scheme.

The first ***raycasting*** algorithm used for generating CG images was fulfilled by A.Apple in 1968. The idea behind this mechanism was to launch a ray from the point of view of the scene, one for each pixel, and try to find the first object that would block its path in the 3D scene. By using the information available in shaders and light sources, it was possible to find out the tonality of the object. One of the advantages offered by ***raycasting*** compared to the old scanline was the fact of being able to take care on non-plane surfaces like spheres and cylinders with great ease. The first real use of ***raycasting*** took place a few years later, in 1966, when a group of mathematicians from the Mathematical Applications Group, Inc (MAGI) used this technology for military purposes, by carrying out calculations on gamma-rays and their effect on the environment. The software projected by the MAGI calculated not only how these rays bounce in the environment, but also how they are absorbed or refracted. These studies were applied for military purposes by the American government, such as radiation-proof vehicle or space-vehicle construction. Later on, in 1968, under the supervision of Dr. Philip Mittelman, scientists developed a new method for generating images with the same software. In 1972 the MAGI became an animation studio. It uses ***raycasting*** for creating 3D animations for commercial and cinematographic use, like for example the famous film TRON. MAGI ceased to exist in 1985.

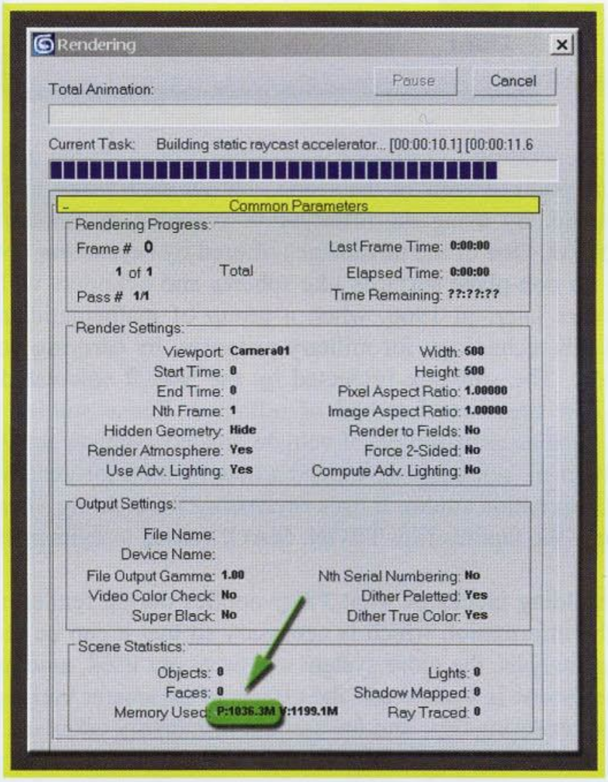
The concepts described so far are the building blocks of what *VRay* applies during rendering calculations. It analyzes the scene and studies it, acquiring all the information which is necessary so that it can be recalled quickly. Obviously, the greater the data accumulated during analysis, the more system resources are used, amongst which the RAM. In the next few examples an averagely complex scene is studied and the effects of parameter variation in the ***raycaster*** section have been analyzed, both with regards to rendering time and the amount of memory taken up.

It is a series of 3,375 spheres, for a total of 7,128,036 triangles. A heavy scene, illuminated without GI through two ***VRayLights***.



■ **Figure 4.315**
The stress burdened by VRay is due to the enormous amount of triangles.

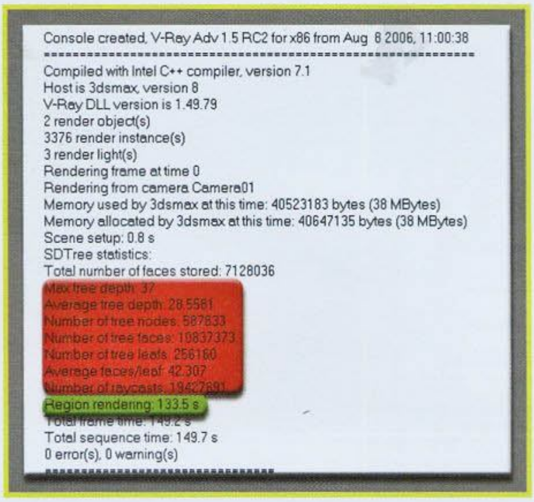
What is interesting is the ratio between the amount of memory taken up by ***VRay*** during rendering and the time taken to render. As far as the first parameter is concerned, it is possible to view RAM use directly through the Renderer panel. In order to obtain rendering time one must input the appropriate data in the ***VFB*** by using ***Frame stamp***.



■ **Figure 4.316**
One can observe the amount of memory used by 3ds Max for rendering directly within the Render Panel.

First of all one can observe how during rendering calculation, ***VRay*** carries out a phase called "***Built static raycast acceleration***". In this phase ***VRay*** is building a structure which contains all the geometrical data used for the actual rendering process. In this phase most of the memory is allocated. By modifying the parameters in the ***Raycaster params*** section, one can control the creation of data structure.

Another important panel which is often ignored is ***VRay messages***. It is a tab associated with every render and contains a lot of useful information, reports and error messages. By analyzing it one can understand certain problems in the scene and get hold of information such as the version of ***VRay***, rendering time, memory usage and information on data structure. Later on, this window will be explained more in-depth. For now it is necessary only to understand a part of this data.



■ **Figure 4.317**
The ***VRay messages*** panel, obtained by the render of the scene being examined. The ***Raycaster*** parameters are emphasized.

Information is found in the following order:

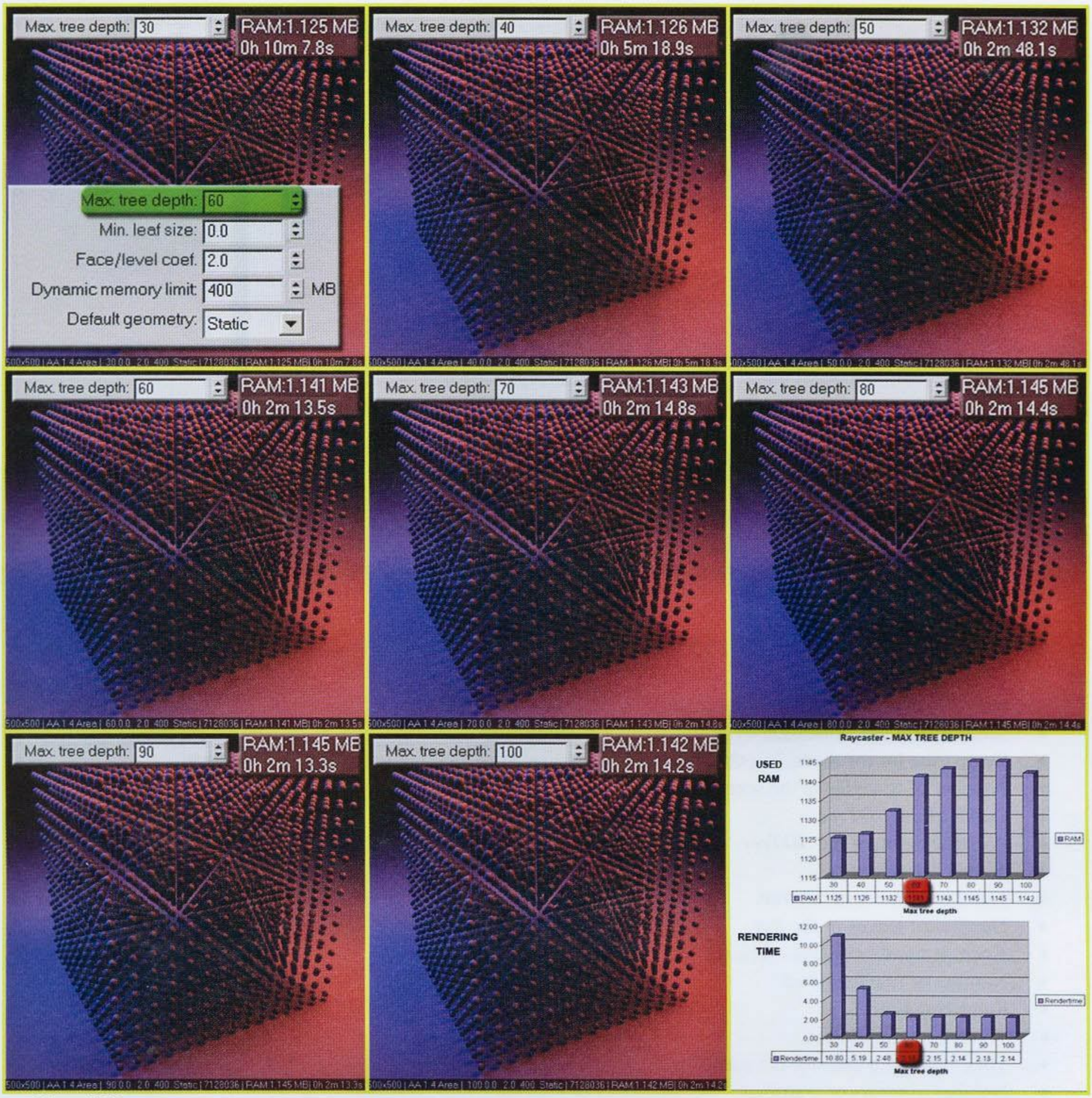
- *VRay* version and date of render creation
- C++ compiler version
- Number of rendered geometrical objects. In our case two: the surface and the sphere which being instances, is perceived as one only.
- Number of instances.
- Number of lights in the scene. 2 ***VRayLights*** and the default light of 3ds Max. Number of rendered frames.
- Name of the rendered frame
- Name of the camera used for rendering.
- Memory use and allocation by 3ds Max.
- Setup time for the scene, in other words the time needed to load the scene into the memory.
- Number of triangles in the scene.
- Depth of the ***SDTree***.
- Average depth of the ***SDTree***.
- Number of nodes.
- Number of faces in the ***tree***.
- Number of leaves in the ***tree***.
- Average number of faces for each leaf.
- Number of ***raycasts*** used.
- Rendering time.
- Total rendering time.
- Total rendering time for the scene.
- Number of errors or warnings.

The parameters which will be observed for now are only those concerning effective rendering time and ***SDTree*** data.

The first series of concerns the ***Raycast params***. This data checks the *VRay* ***BSP tree***. One of the main and most important operations that *VRay* can carry out at the beginning of a rendering calculation is the so-called ***raycasting***. With this passage *VRay* emits a series of Rays from the observer's point of view and decides whether these rays intersect any geometries. If this happens it identifies the object and stores the relevant information. Obviously testing if a ray intersects all the faces in a scene, in this case 7 million, is very complex and takes a long time. In order to speed up the calculation the concept of ***BSP tree*** is therefore introduced, a hierarchic tree structure.

The parameters which allow the tree to be managed are the following:

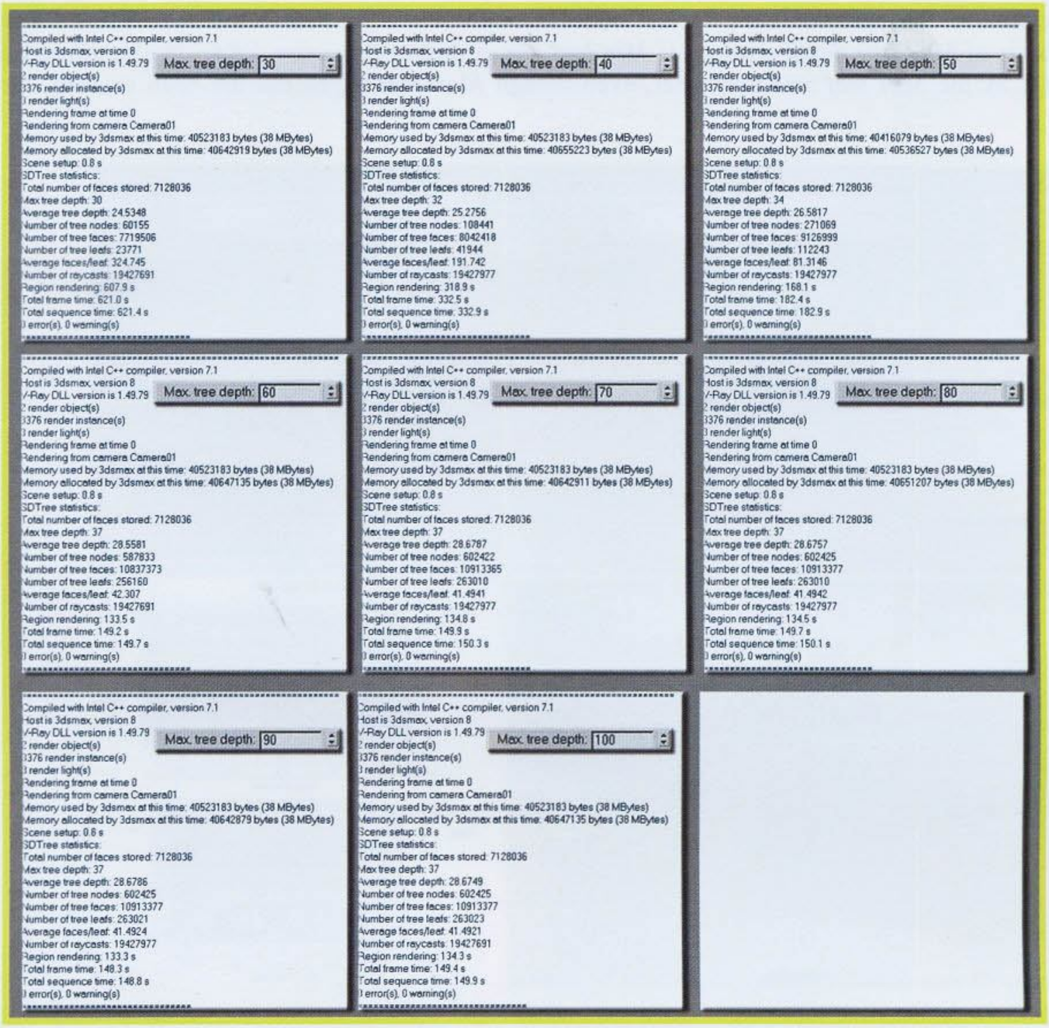
Max tree depth - it allows tree depth to be managed. The higher this value is, the greater the number of branches. This requires additional memory, but can make rendering faster. The lower its value, the less RAM is used, at the expense of rendering time. It must be remembered that there is no single value for each scene. The presence of ***Displacement*** or ***VRayFur*** of single objects can define optimal values for ***Max tree depth*** which differ.



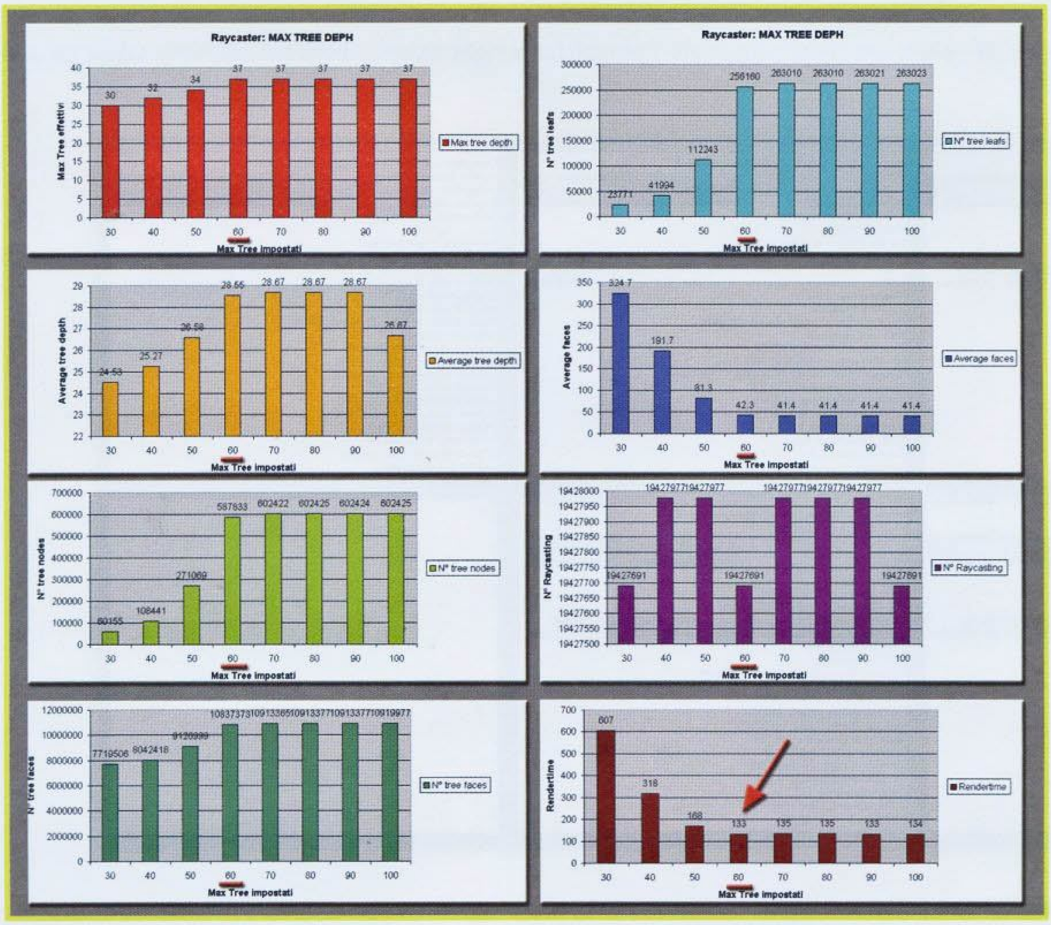
■ **Figure 4.318**
Effects on rendering time and memory taken up by VRay due to parameter variation of ***Max tree depth***. All the other parameters are default.

In this particular scene the fact of lowering ***Max tree depth*** to a value of 30 has led to a high rendering time and a slight lowering in RAM usage. Moreover, values above 60 do not lower rendering time in any way. Beyond this threshold, the time taken is around 2 min. 14 sec.

For this scene the default value of 60 is optimal. One can state this after having analyzed the ***VRay message*** report. With use of the information contained in this window, one notices how most output concerning the ***SDTree*** become constant above 60.

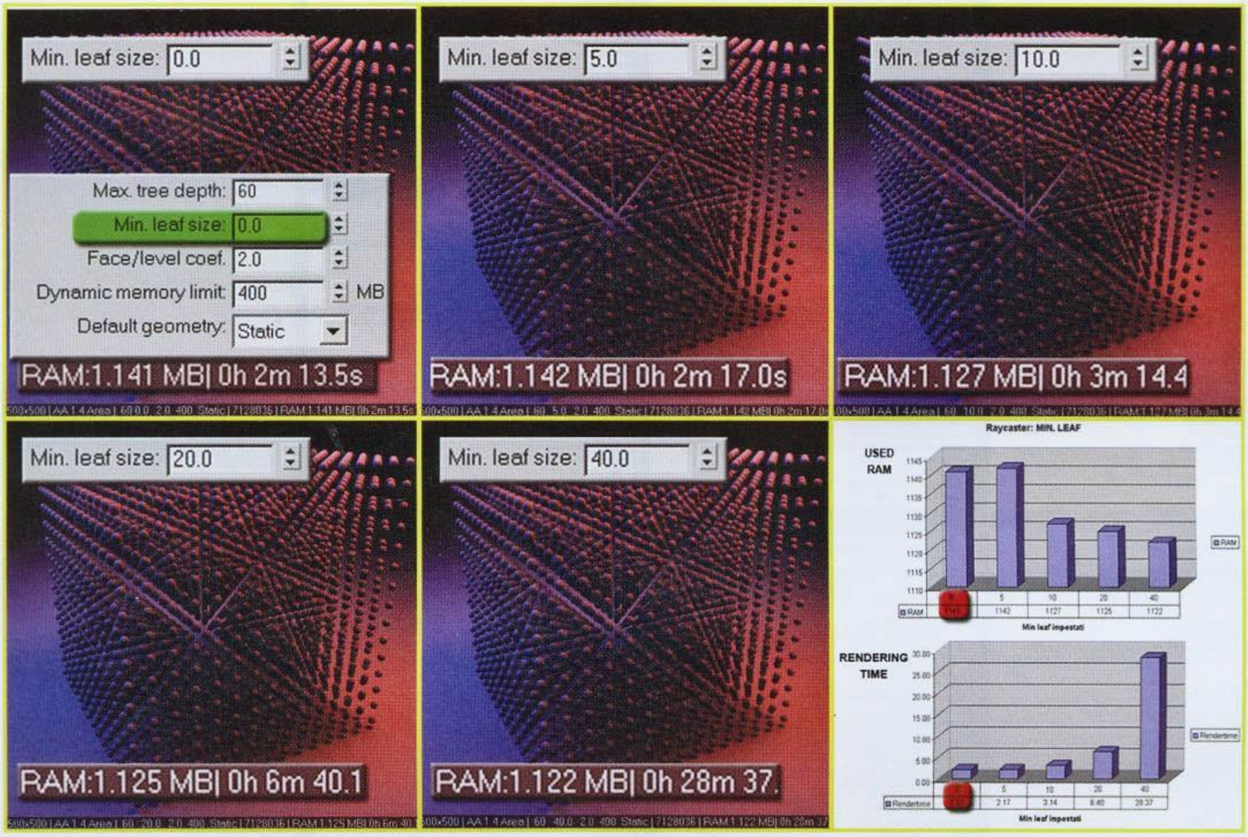


■ **Figure 4.319**
The eight tabs in ***VRay message***.



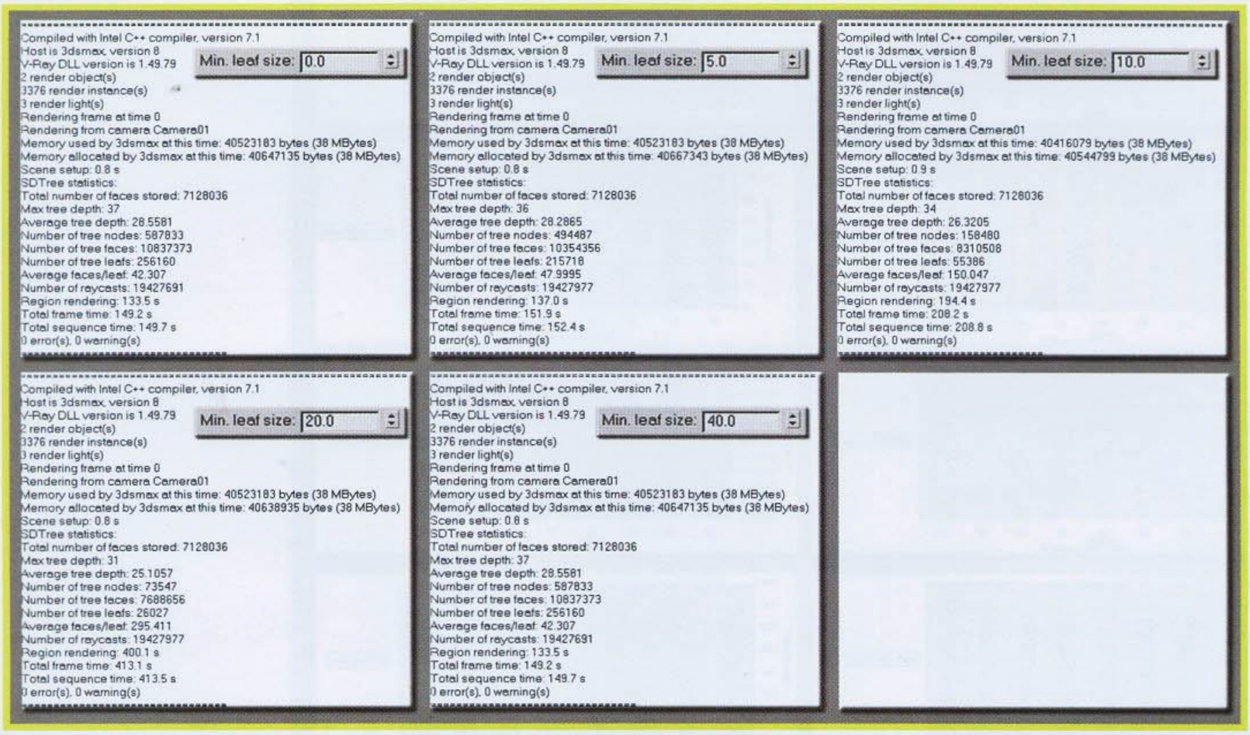
■ **Figure 4.320**
Eight graphics fulfilled with the data in ***VRay message***.

Min leaf size - this determines the minimum size for leaf nodes. Usually this is set to 0.0: this means that *VRay* divides the scene on the basis of ***Max tree depth*** settings. If ***Min leaf size*** is greater than zero, *VRay* divides the scene until it reaches the size set by the ***Min leaf size*** parameter, even though ***Max tree depth*** has not been reached, for example.

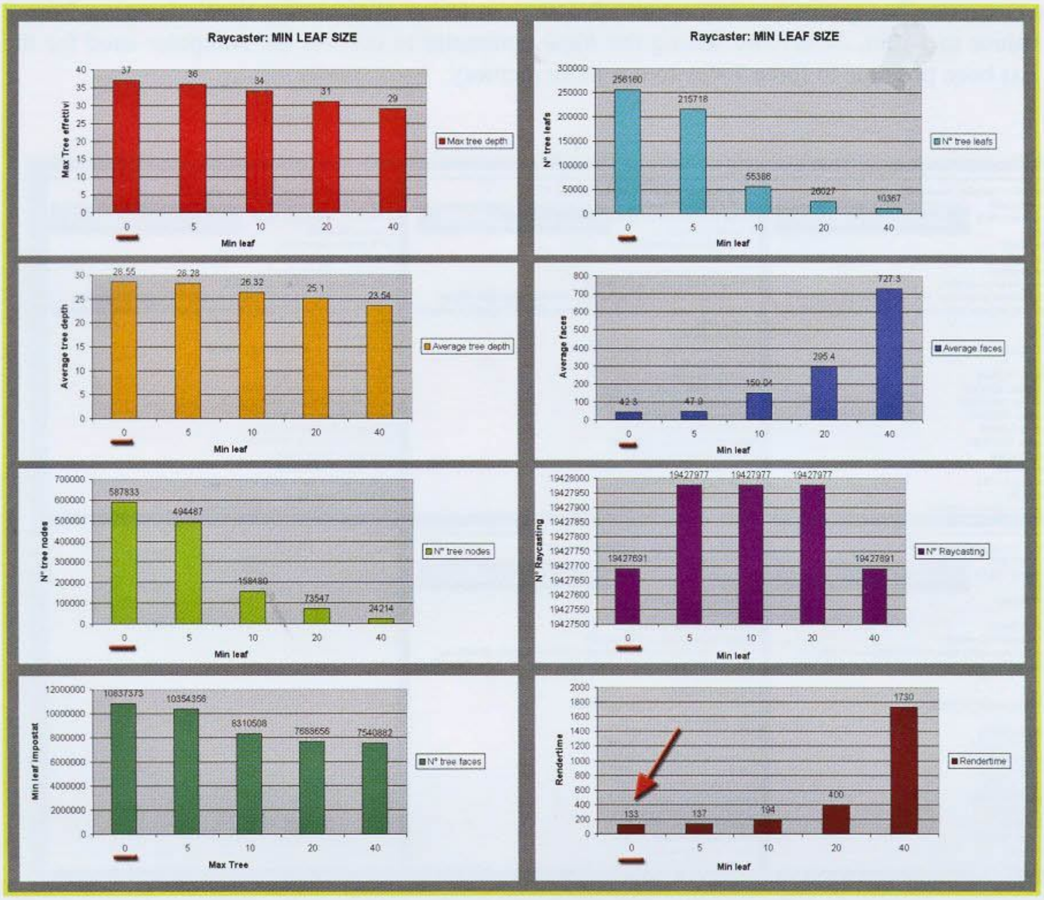


■ **Figure 4.321**
Effects on rendering time and memory used by VRay due to ***Min leaf*** parameter variations.

As in the previous case, default values are those which are optimal for a right balance between memory taken up and rendering time.

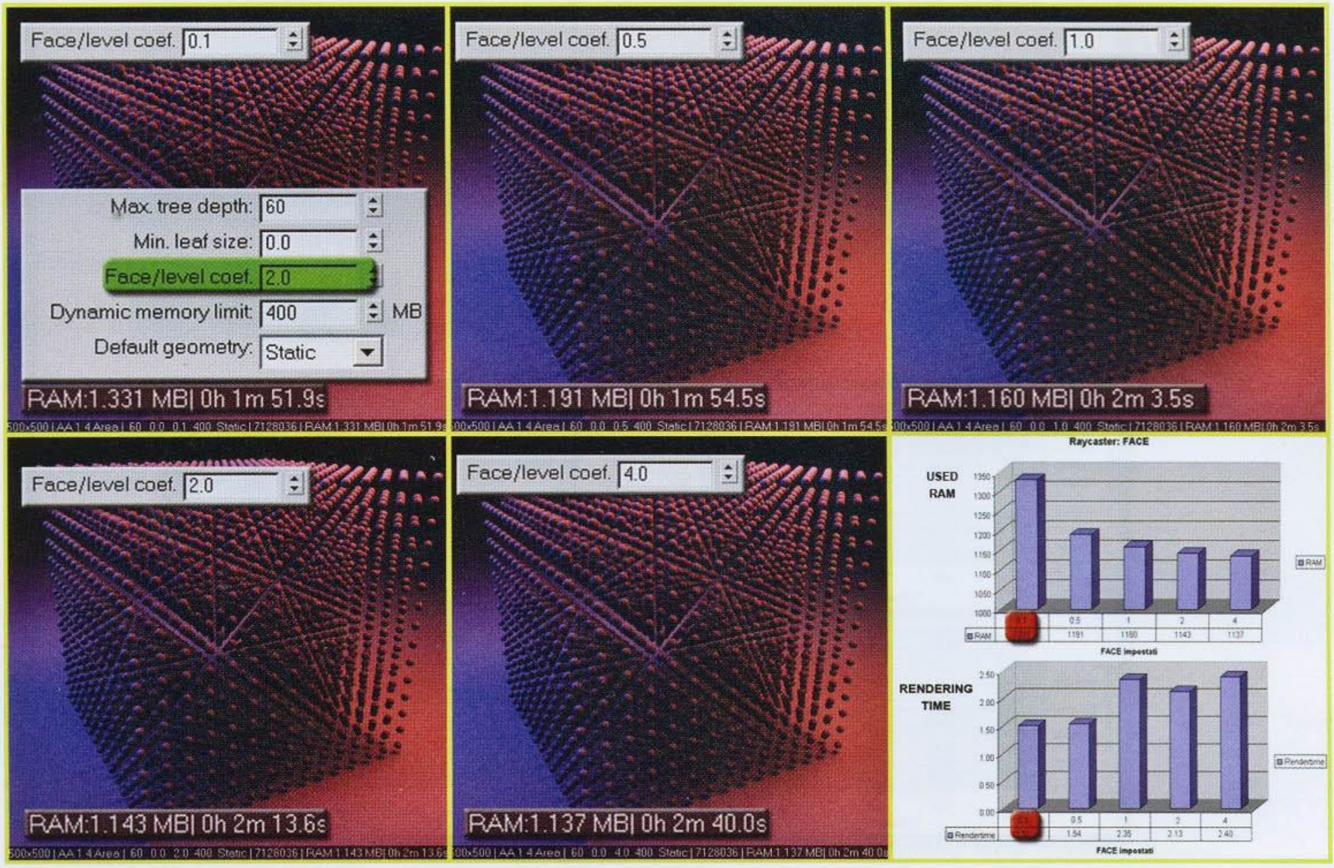


■ **Figure 4.322**
VRay message obtained from tests which have just been carried out.



■ **Figure 4.323**
Graphics created with the data given by ***VRay message***.

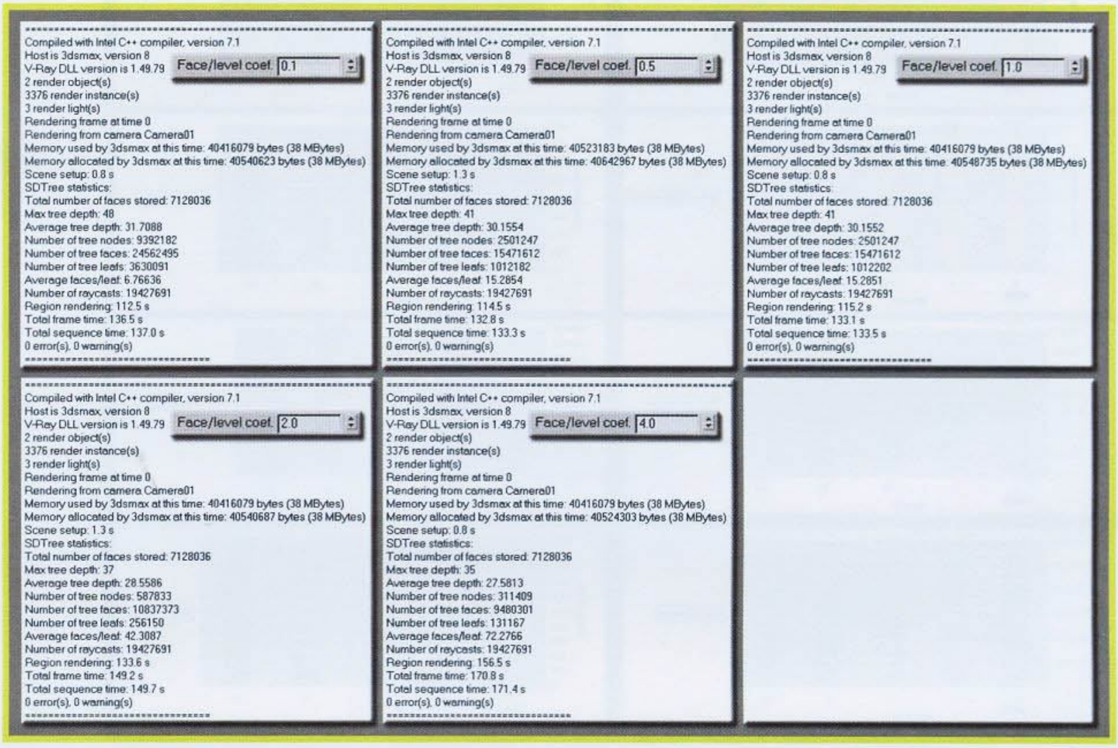
Face/level coef. - it controls the maximum number of triangles which can be found in each leaf node. The smaller this value is, the faster rendering becomes, at the expense of memory. Also in this case, each scene has different optimal setting values.



■ **Figure 4.324**
Effects on rendering time and memory usage due to the variation in the ***Face/level coef.***

In this case decreasing the parameter has granted an important advantage in terms of rendering time, changing from 2 min. 13 sec. with default values to 1 min. 51 sec. by setting the ***Face*** parameter to 0.1. As the computer used for the tests has 2 GB of RAM, it has been possible to force ***VRay*** to use more memory.

■ **Figure 4.325** ***VRay message*** obtained from the tests which have just been carried out, by modifying the ***Face/level coef.***



```
Compiled with Intel C++ compiler, version 7.1
Host is 3dsmax, version 8
V-Ray DLL version is 1.49.79    Face/level coef. 0.1
2 render object(s)
3376 render instance(s)
3 render light(s)
Rendering frame at time 0
Rendering from camera Camera01
Memory used by 3dsmax at this time: 40416079 bytes (38 MBytes)
Memory allocated by 3dsmax at this time: 40540623 bytes (38 MBytes)
Scene setup: 0.8 s
SDTree statistics:
Total number of faces stored: 7128036
Max tree depth: 48
Average tree depth: 31.7088
Number of tree nodes: 9392182
Number of tree faces: 24562495
Number of tree leafs: 3630091
Average faces/leaf: 6.76636
Number of raycasts: 19427691
Region rendering: 112.5 s
Total frame time: 136.5 s
Total sequence time: 137.0 s
0 error(s), 0 warning(s)
```

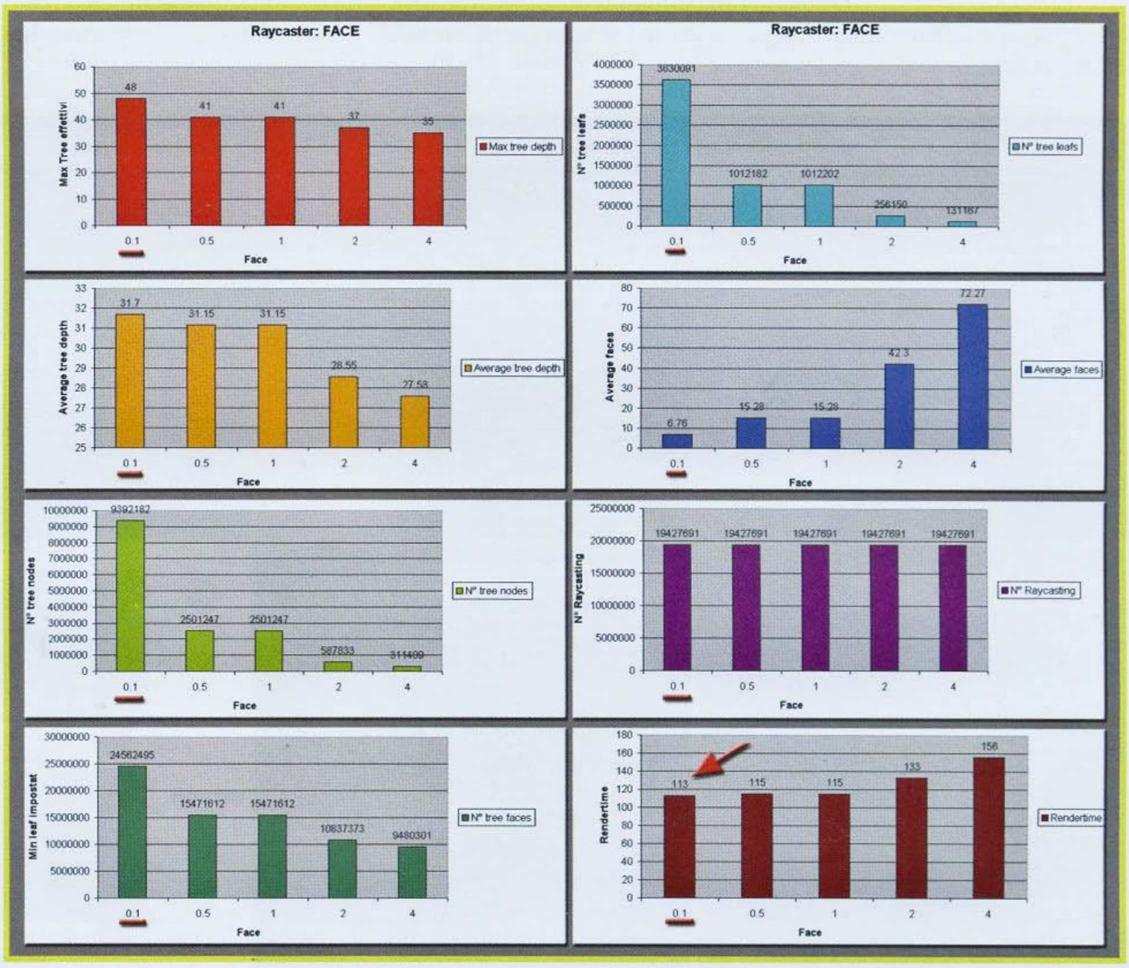
```
Compiled with Intel C++ compiler, version 7.1
Host is 3dsmax, version 8
V-Ray DLL version is 1.49.79    Face/level coef. 0.5
2 render object(s)
3376 render instance(s)
3 render light(s)
Rendering frame at time 0
Rendering from camera Camera01
Memory used by 3dsmax at this time: 40523183 bytes (38 MBytes)
Memory allocated by 3dsmax at this time: 40642967 bytes (38 MBytes)
Scene setup: 1.3 s
SDTree statistics:
Total number of faces stored: 7128036
Max tree depth: 41
Average tree depth: 30.1554
Number of tree nodes: 2501247
Number of tree faces: 15471612
Number of tree leafs: 1012182
Average faces/leaf: 15.2854
Number of raycasts: 19427691
Region rendering: 114.5 s
Total frame time: 132.8 s
Total sequence time: 133.3 s
0 error(s), 0 warning(s)
```

```
Compiled with Intel C++ compiler, version 7.1
Host is 3dsmax, version 8
V-Ray DLL version is 1.49.79    Face/level coef. 1.0
2 render object(s)
3376 render instance(s)
3 render light(s)
Rendering frame at time 0
Rendering from camera Camera01
Memory used by 3dsmax at this time: 40416079 bytes (38 MBytes)
Memory allocated by 3dsmax at this time: 40548735 bytes (38 MBytes)
Scene setup: 0.8 s
SDTree statistics:
Total number of faces stored: 7128036
Max tree depth: 41
Average tree depth: 30.1552
Number of tree nodes: 2501247
Number of tree faces: 15471612
Number of tree leafs: 1012202
Average faces/leaf: 15.2851
Number of raycasts: 19427691
Region rendering: 115.2 s
Total frame time: 133.1 s
Total sequence time: 133.5 s
0 error(s), 0 warning(s)
```

```
Compiled with Intel C++ compiler, version 7.1
Host is 3dsmax, version 8
V-Ray DLL version is 1.49.79    Face/level coef. 2.0
2 render object(s)
3376 render instance(s)
3 render light(s)
Rendering frame at time 0
Rendering from camera Camera01
Memory used by 3dsmax at this time: 40416079 bytes (38 MBytes)
Memory allocated by 3dsmax at this time: 40540687 bytes (38 MBytes)
Scene setup: 1.3 s
SDTree statistics:
Total number of faces stored: 7128036
Max tree depth: 37
Average tree depth: 28.5586
Number of tree nodes: 587833
Number of tree faces: 10837373
Number of tree leafs: 256150
Average faces/leaf: 42.3087
Number of raycasts: 19427691
Region rendering: 133.6 s
Total frame time: 149.2 s
Total sequence time: 149.7 s
0 error(s), 0 warning(s)
```

```
Compiled with Intel C++ compiler, version 7.1
Host is 3dsmax, version 8
V-Ray DLL version is 1.49.79    Face/level coef. 4.0
2 render object(s)
3376 render instance(s)
3 render light(s)
Rendering frame at time 0
Rendering from camera Camera01
Memory used by 3dsmax at this time: 40416079 bytes (38 MBytes)
Memory allocated by 3dsmax at this time: 40524303 bytes (38 MBytes)
Scene setup: 0.8 s
SDTree statistics:
Total number of faces stored: 7128036
Max tree depth: 35
Average tree depth: 27.5813
Number of tree nodes: 311409
Number of tree faces: 9480301
Number of tree leafs: 131167
Average faces/leaf: 72.2766
Number of raycasts: 19427691
Region rendering: 156.5 s
Total frame time: 170.8 s
Total sequence time: 171.4 s
0 error(s), 0 warning(s)
```

■ **Figure 4.326** Graphics fulfilled with the data found in ***VRay message***.

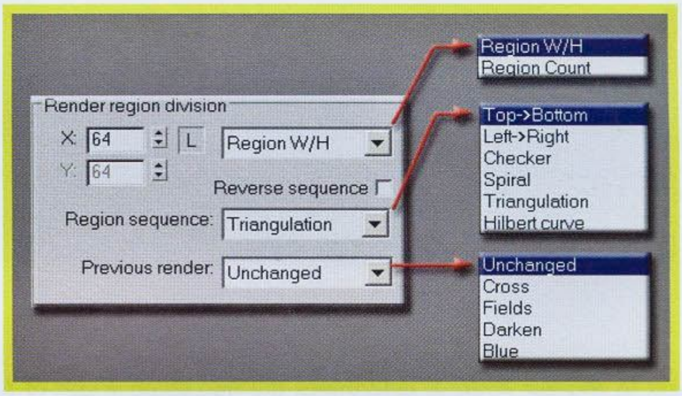


As far as the subject of static and dynamic geometry is concerned, as it is quite complex, we will limit ourselves to practical explanations. The main difference between these two types regards the way *VRay* treats system memory. If one uses the static mode at the beginning of a render, *VRay* loads all the geometrical objects in a scene into the memory and therefore also data structure which will remain there until the end of rendering. Instead by using the dynamic system, *VRay* loads and downloads from the memory only the data needed for the portion of a frame which is being calculated. If in the upper part of the render there are tens of geometrically modelled trees, *VRay* requires much more RAM. Vice versa, if the lower part is occupied only by a simple plane, more memory will be taken up when *VRay* buckets (small square portions which are developed in succession during rendering) are placed on the layer of vegetation. This memory is freed up when the plane is rendered.
As far as elements such as ***VRayFur***, ***VRayProxy*** and ***Displacement*** are concerned, *VRay* considers them as dynamic geometries. This means that even if one has chosen a static system in the settings, these objects are always treated as dynamic elements.

Dynamic memory limit - this determines the maximum quantity of RAM available for ***Raycaster*** when in ***Dynamic*** mode. It is important to specify that this quantity, when using a multi-CPU system, is divided between the processors. For example, if we are using a CPU Core2Duo each processor uses half the RAM made available. Memory limits, if set too low, can cause important renders to lag, especially if geometries are to be continuously loaded and unloaded.

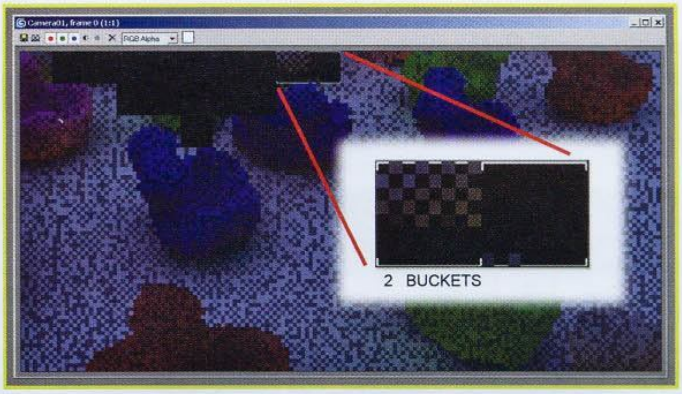
Render region division

The section called ***Render region division*** manages the way *VRay* generates renders within the ***Frame buffer*** both in 3ds Max and proprietary software.



■ **Figure 4.327**
The section dedicated to ***bucket*** management. The three pull-down menus with relative parameters can be seen here.

One of the things which can be noticed when one first starts up a render with *VRay* is the presence of a small square which moves about inside the ***Frame buffer*** during image calculation. This square is called bucket and the ***Render region division*** has the purpose of controlling its parameters.



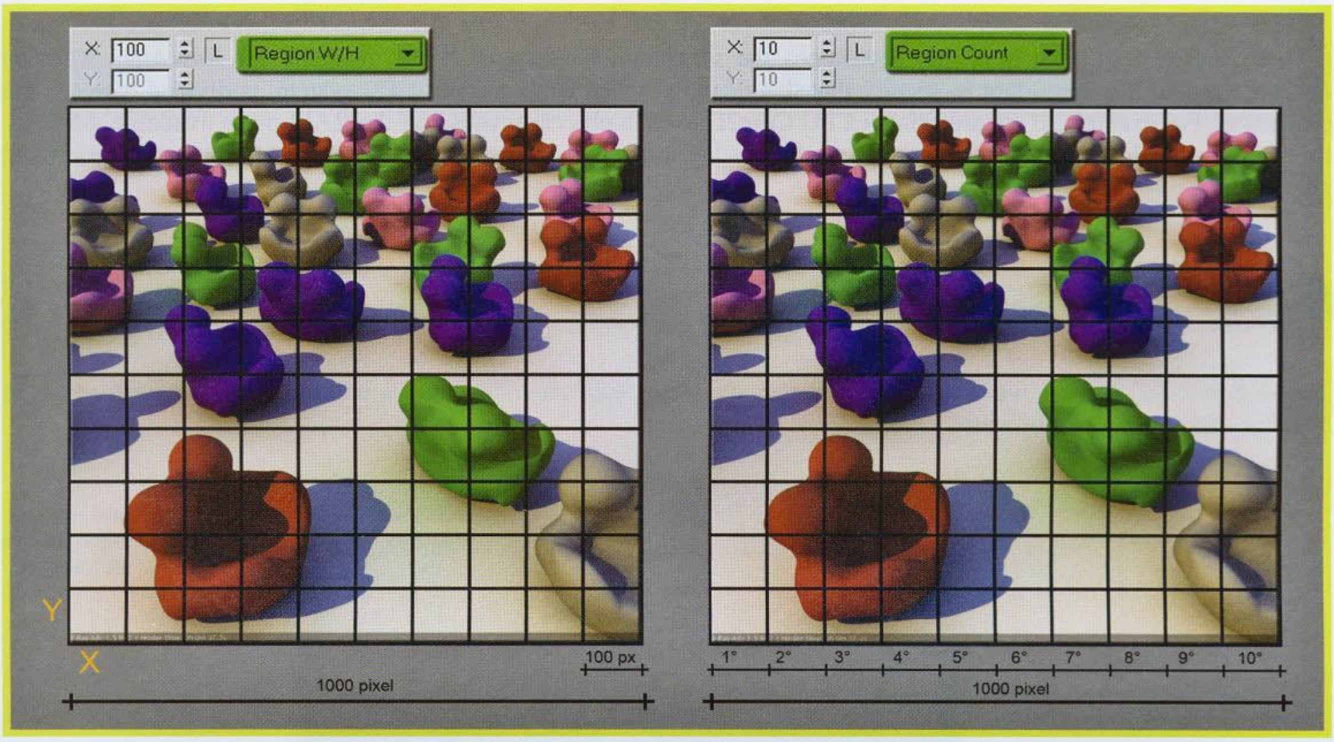
■ **Figure 4.328**
The computer is endowed with a CPU DualCore. This is why there are two ***buckets***, one for each thread. If we were using a quad core CPU there would be four ***buckets***.

The term ***bucket*** indicates a rectangular portion of the render which can be directly computed independently. Compared to the traditional rendering method in 3ds Max which is Scanline and renders by means of horizontal lines, *VRay* subdivides the image to be rendered through a grid.
The squares which are obtained (***buckets***) can be distributed amongst more than one core of the same CPU or on more than one computer via a distributed rendering system made available by *VRay*.

Buckets which are too small can slow down the rendering process, as *VRay* would have to compute the union between the various portions many more times, especially in a distributed render. One can reduce rendering time by increasing their size. This way *VRay* will use more RAM however, as the portion of rendering to be calculated is greater.

Region W/H - *VRay* determines the size (height and width) of ***buckets*** by using pixels as the measuring unit. If rendering output is 1000 x 1000 pixels and one uses ***buckets*** with a size equivalent to 100 x 100 pixels, *VRay* subdivides the image into 100 portions.

Region Count - *VRay* determines the number of square portions according to the number set by the user in the X and Y parameters. If yet again we take the example of an image with 1000 x 1000 pixels, if we set the X and Y value to 10, *VRay* subdivides the image into 100 ***buckets***.

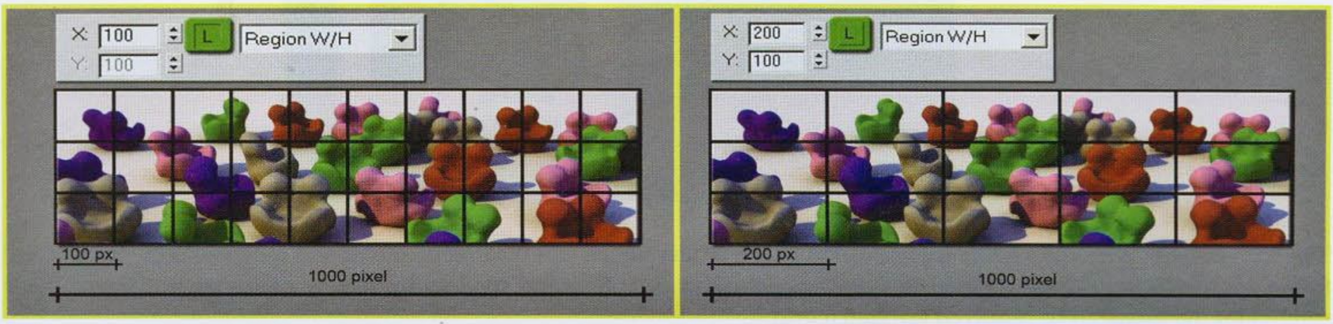


■ **Figure 4.329**
Bucket subdivision.

X - if the subdivision is set to ***Region W/H***, X determines the maximum width of ***buckets*** expressed in pixels. Vice versa, if ***Region Count*** is selected it determines the number of horizontal subdivisions.

Y - if the subdivision is set to ***Region W/H***, X determines the maximum height of ***buckets*** expressed in pixels. Vice versa, if ***Region Count*** is selected it determines the number of vertical subdivisions.

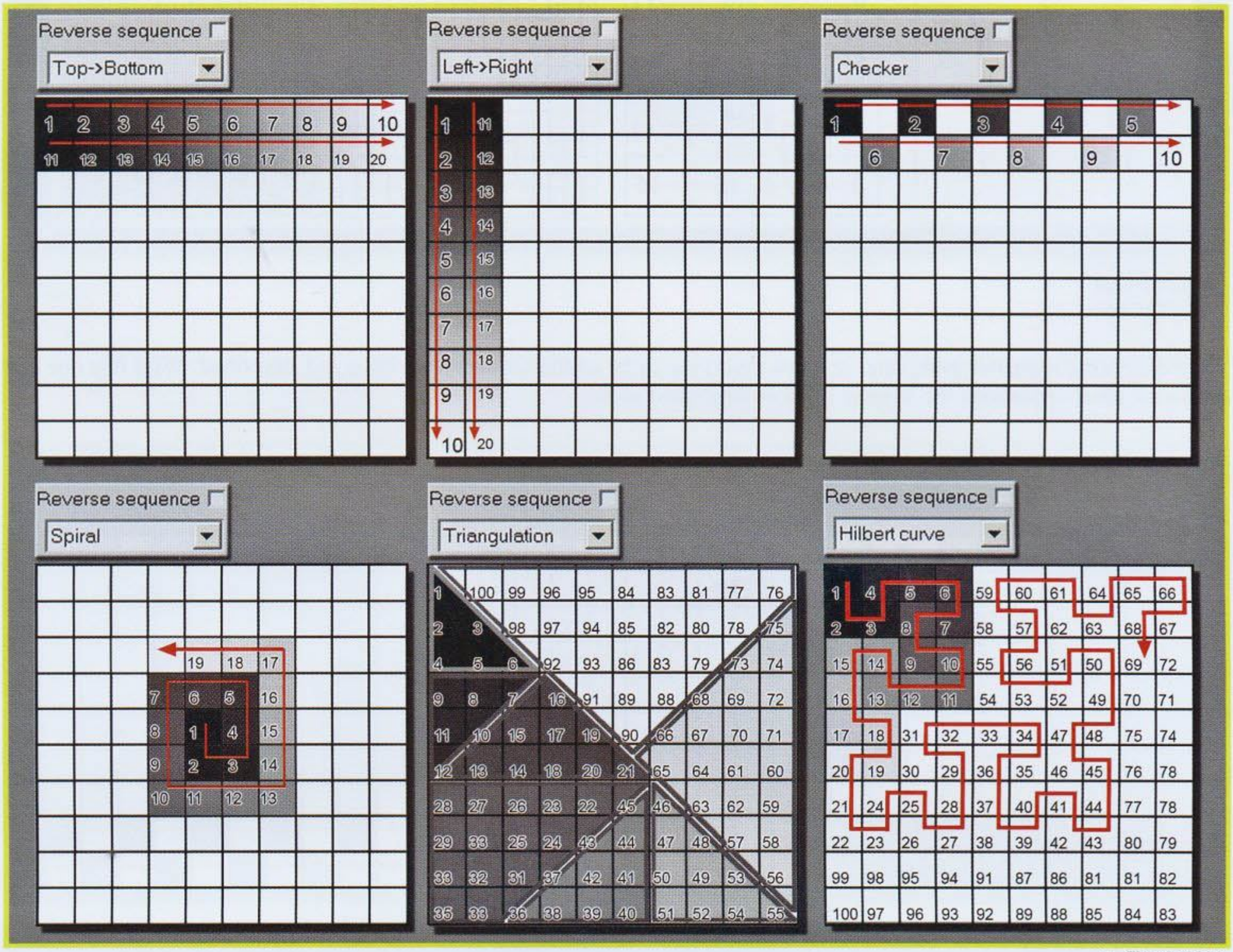
L - the small button resembling an L (***lock***) allows one to lock the value of Y as a function of X. In other words, by changing X, Y is automatically changed. This allows one, for example, to obtain perfectly square ***buckets***, both in ***Region W/H*** mode and ***Region Count*** mode. By disabling it, the ratio between X and Y can be different than 1.



■ **Figure 4.330**
Buckets of different sizes.

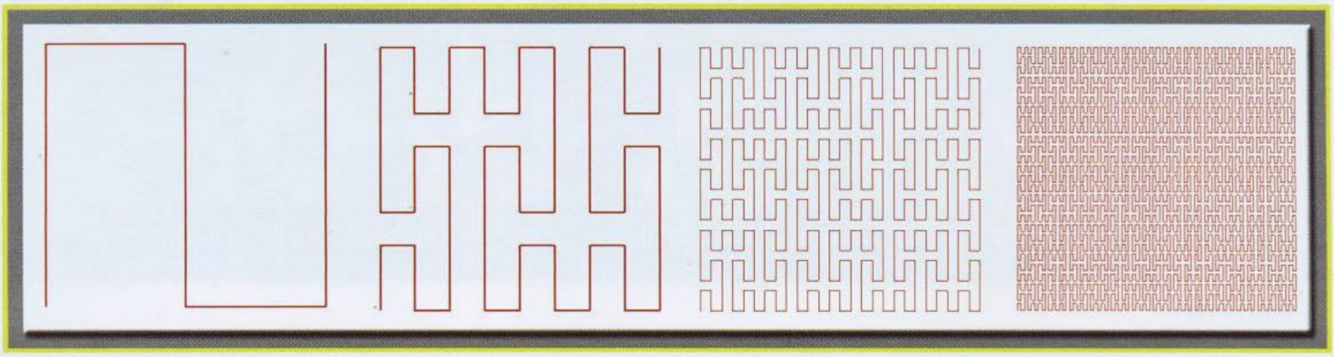
As well as the size of ***buckets***, one can decide the order or sequence which ***buckets*** should follow during rendering. For example one can decide to render from starting from the upper part downwards, from the centre towards the outside and so on. There are six rendering orders.

. Top/botton
. Left/right
. Checker
. Spiral
. Triangulation
. Hilbert curve



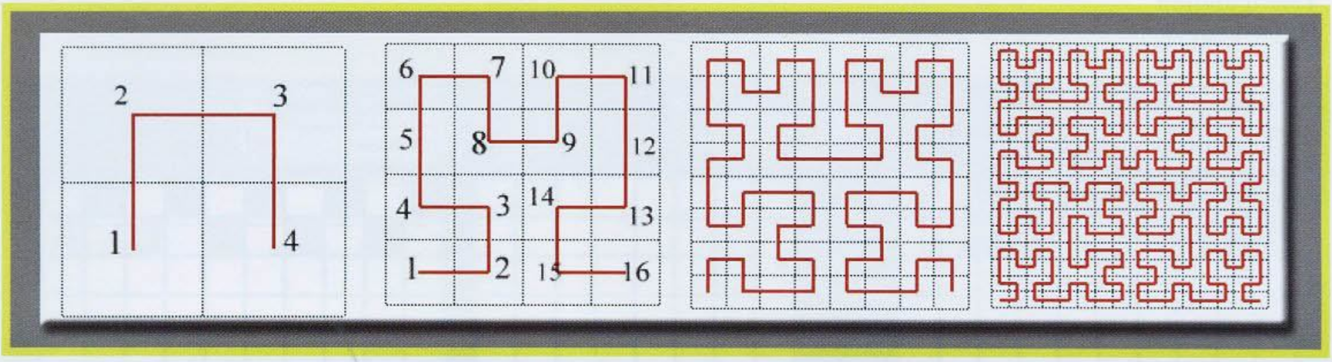
■ Figure 4.331
The six sequences available for rendering with ***buckets***.

The last sequence is of particular interest. It is a curve called Hilbert curve. This dates back to 1890 when the Italian mathematician Giuseppe Peano (1858-1932) published an article with the title "Sur une courbe qui remplit toute une aire plan" which represented a curve which had the strange property of filling a square.



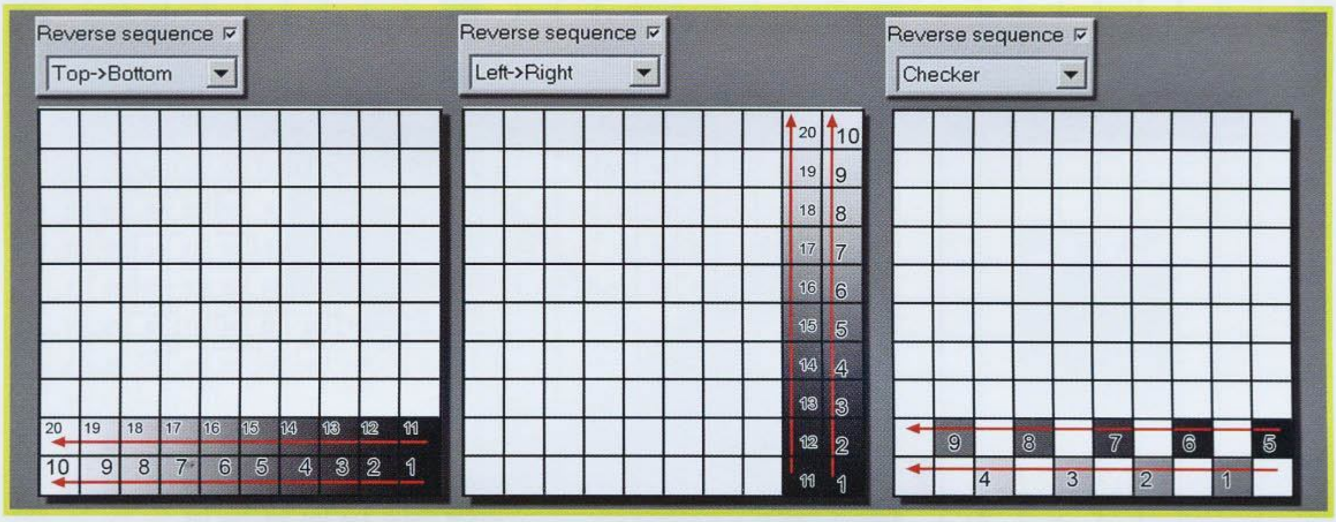
■ Figure 4.332
Peano's curve.

This fact caused a certain amount of perplexity as a curve, by definition, is a geometrical element in one dimension whereas a square has two dimensions, and yet Peano's curve passes through every point in the square.
Another example, similar to Peano's curve, was presented in 1981 by David Hilbert. His idea was to divide the square into 4 parts which were then numbered, each one corresponding to a quarter of the starting sequence; the procedure is repeated for each quarter and the sixteen squares which are thus obtained are numbered. As each one corresponds to one sixteenth of the starting segment, numbering must be consecutive, meaning that one can only move from one square to a neighbouring one. In order to obtain an approximation, each time closer to the curve suggested by Hilbert, one must apply this process an infinite number of times.



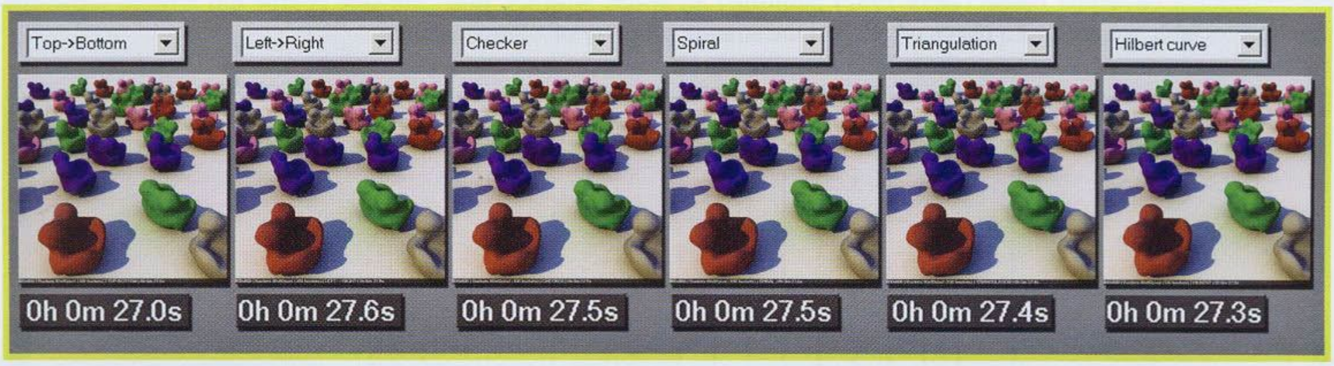
■ **Figure 4.333**
Hilbert's curve.

Via the ***Reverse sequence*** parameter one can invert the order of the sequence we have just described. With this one can obtain six other sequences, the inverse of those described so far.



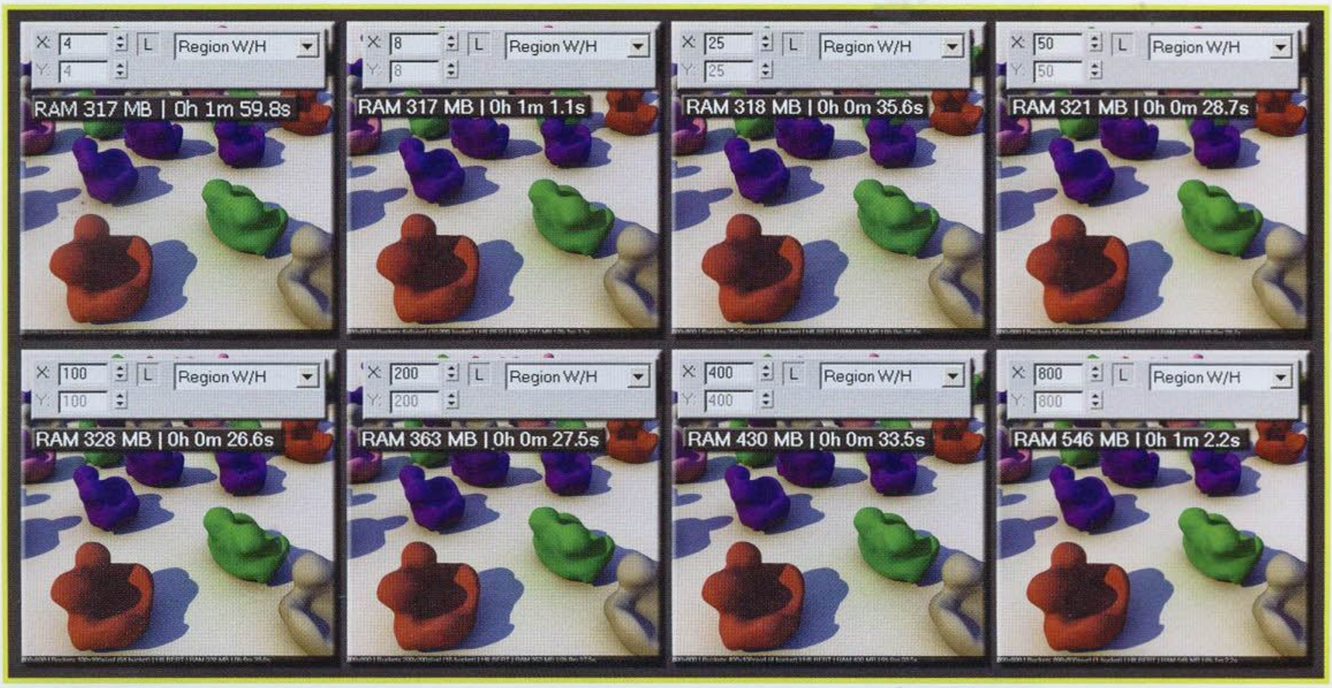
■ **Figure 4.334**
Alcune sequenze invertite.

As far as rendering times are concerned, in the scene examined there is absolutely no difference between one sequence and another.



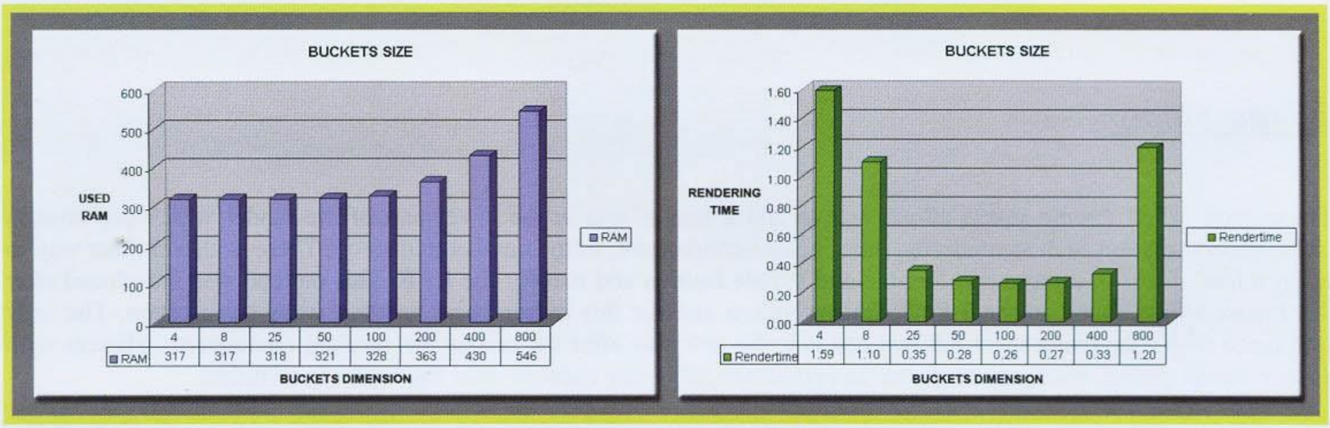
■ **Figure 4.335**
Effect on the calculation sequence following various rendering sequences.

Vice versa, the size of ***buckets*** influences both rendering time and the memory taken up.



■ **Figure 4.336**
The effects of ***bucket*** size on both rendering time and memory usage for an image of 800x800 pixels.

In this scene the best time/RAM ratio is obtained with ***buckets*** of a size of 100x100 pixels. The memory taken up is 328 MB, only 11 Mb more than the minimum, obtained with ***bucket*** size set to 4x4 with a lower time than all the other tests.



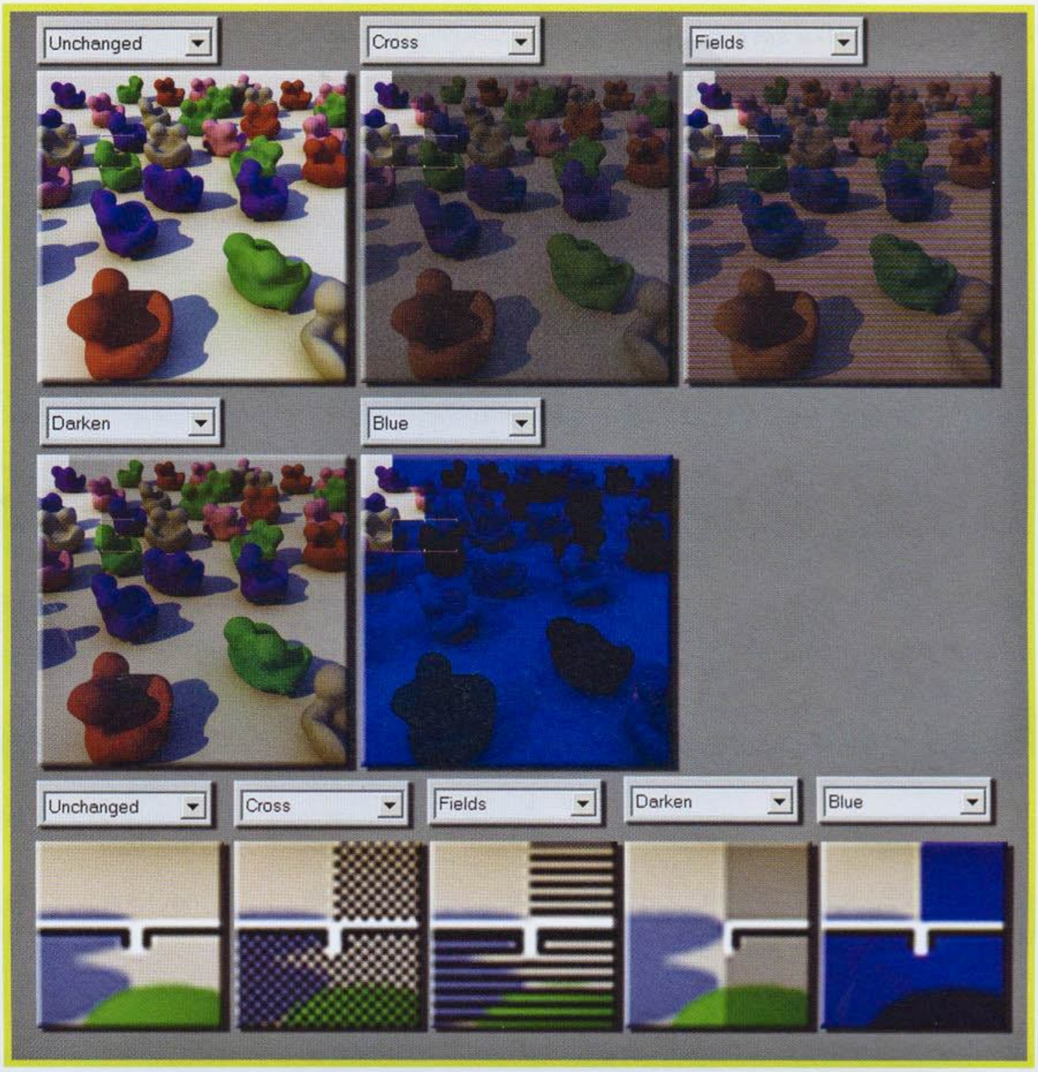
■ **Figure 4.337**
Graphic concerning previous tests. Values between 50 and 100 pixels are the most correct ones.

Previous render - with this pull-down menu one can establish the way images should be handled when within the ***Frame buffer*** when fulfilling a new render. *VRay* offers 5 options:

Unchange : nothing is modified. The ***Frame buffer*** remains unaltered.
Cross : pixels n° 2,4,6,8, etc.. from the whole ***Frame buffer*** are cancelled.
Fields : each line of even pixels is erased.
Darken : the image of the ***Frame buffer*** is darkened.
Blue : the image of the ***Frame buffer*** becomes blue.

These modes do not influence the quality or time taken for rendering. They are exclusively for visual support in order to have a simpler view of what *VRay* is calculating.

■ Figure 4.338
The 5 modes for superimposition.



Frame stamp

The section called ***Frame stamp*** allows one to add a line of text in the lower part of the render which can contain technical information such as rendering time, ***VRay*** version used, frame number and so on. There is also another way to insert a line of text in a more comfortable and simple fashion and namely the ***VFB***. This method was introduced after the ***Frame stamp*** found in the ***VRay: System*** rollout and for this reason there are two ways of insertion. The only difference is that by using the ***VFB*** one can edit the text also after the render has reached completion, whereas with ***Frame stamp system***, once activated, one can no longer make any changes once the render is complete.

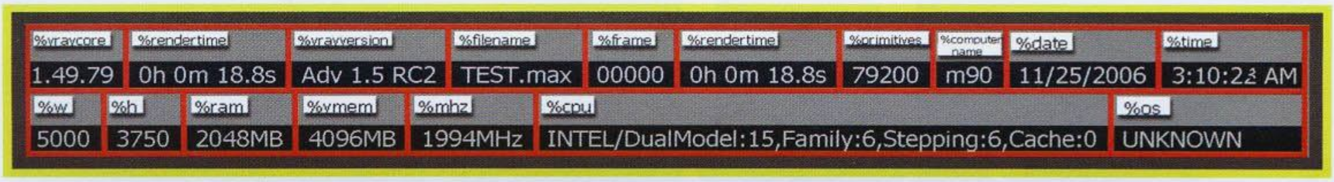
■ Figure 4.339
The ***Frame stamp*** section.

Once rendering has been launched, by clicking on the appropriate checkbox one can edit the string of text both through conventional words or keywords with the prefix ***%***.

%vraycore	: n° of the *VRay* core version.
%rendertime	: time taken for the last render.
%vrayversion	: internal version of *VRay*.
%filename	: filename.
%frame	: n° of the current frame.
%primitives	: number of triangles used in the render.
%computername	: name of the computer.
%date	: date of render completion.
%time	: time of computer completion.
%w	: width of the render measured in pixels.
%h	: height of the render measured in pixels.
%ram	: physical memory available.
%vmem	: virtual memory available.
%mhz	: processor speed measured in MHz.
%cpu	: type of CPU installed in the system.
%os	: OS version.

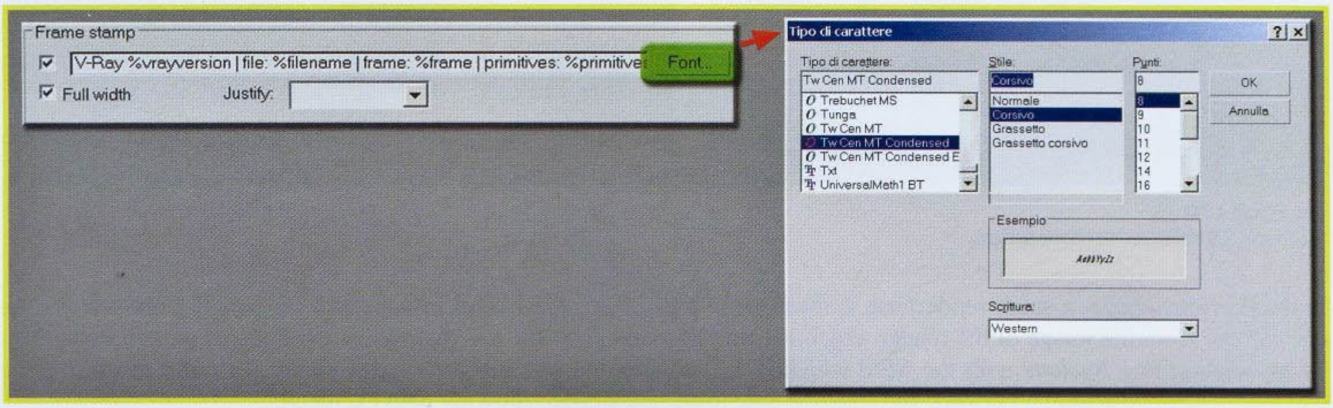
■ **Figure 4.340**
Chart with all the possible variables which can be used within the ***Frame stamp***.

The following image is an example with all the variables inserted.



■ **Figure 4.341**
Test with all the variables activated. The version 1.5 RC2 does not recognize Microsoft OS X64. This is why UNKNOWN appears.

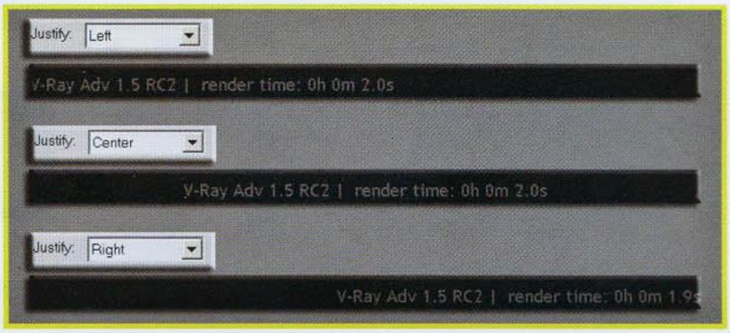
Font - with this command one can recall the Windows "Character type" window, which enables one to change font, style and size of the text within ***Frame stamp***.



■ **Figure 4.342**
Activation of the ***Font*** window.

Full width - with this parameter activated, the ***Frame stamp*** occupies the whole width of the image.

Justify - this modifies the position of the text within ***Frame stamp***, by moving it to the left, right or centre.



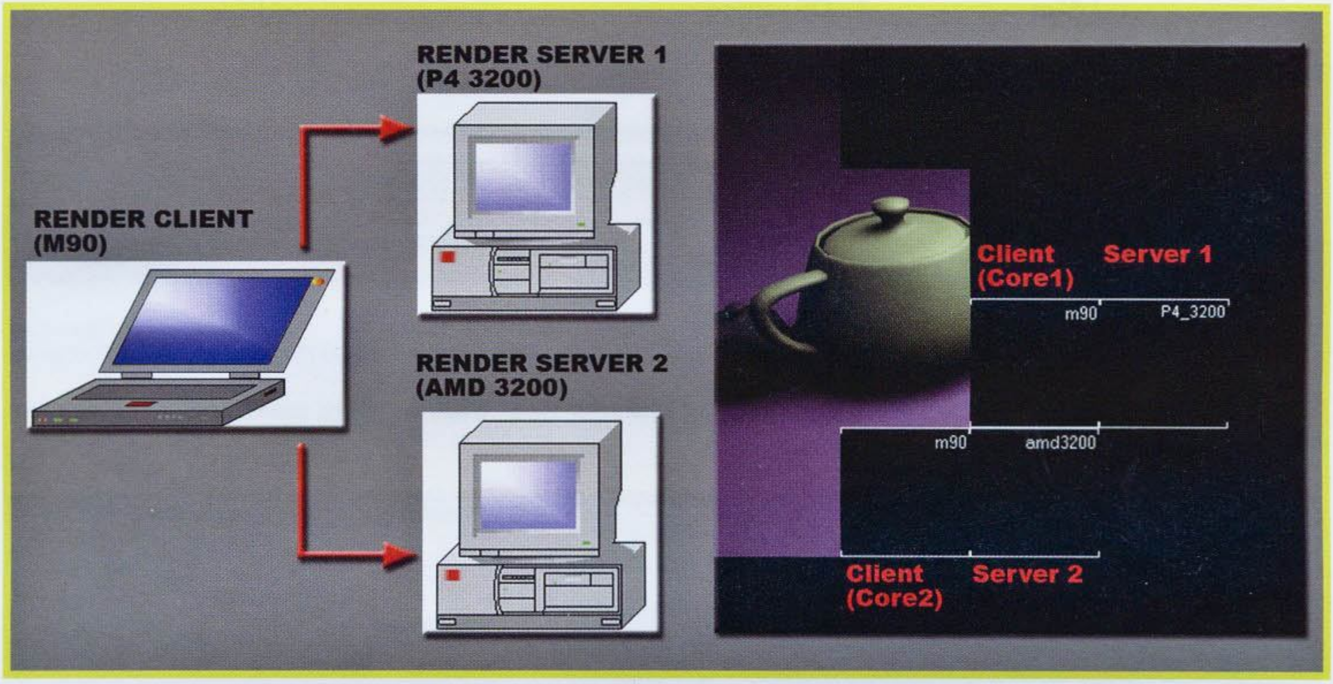
■ **Figure 4.343**
The three variables of the ***Justify*** parameter.

Distribuited rendering

Distributed rendering is a technique for spreading the render among different computers. There are many ways to undergo this procedure, but its main use is to reduce rendering times by assigning the frame, the image itself, to computers in a network. The method consists of subdividing the render into the afore mentioned ***buckets*** and then making each computer take care of a different ***bucket***, by distributing them via TCP/IP protocol. The two main figures in this system are ***Render Clients*** and ***Render Servers***.

Render client - this is the machine where the file to be rendered is loaded and where the command for rendering is launched from. It is the duty of this PC to command the others, subdivide the frame into ***buckets*** and distribute these to the other PCs. All data will be gathered on this PC and the render will be saved here.

Render servers - these are the computer where the render client distribuites the various ***buckets***. A group of ***Render servers*** is called renderfarm.

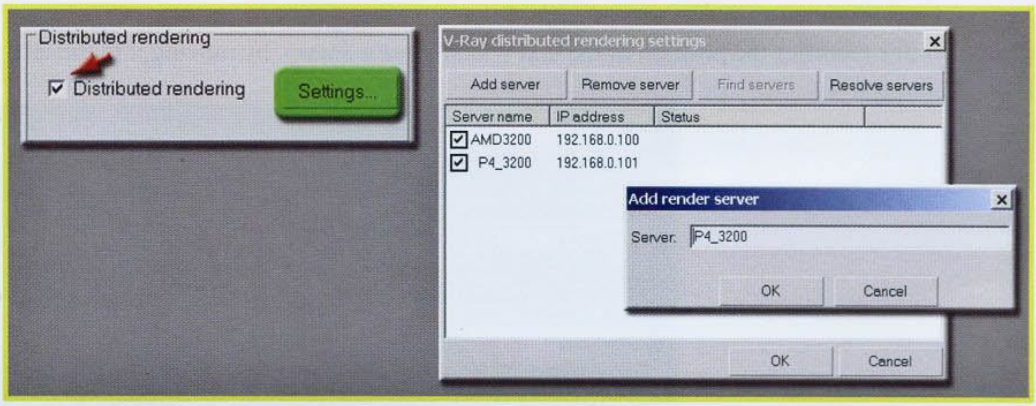


■ **Figure 4.344**
Scheme of a distributed rendering system.

In the previous image, a small renderfarm is made up of a PC client M90 Dualcore and two PC Single Core servers. As one can see from the image of the render which is being carried out, *VRay* has distributed the ***buckets*** on the renderfarm by associating two ***buckets*** with the M90 which is a Dualcore and one ***bucket*** to each server. In order to do this the computers must be within a network and must be able to see each other allowing data exchange.

In order to launch ***Distributed rendering*** one must activate the relevant checkbox. After one must configure parameters by clicking the ***Settings*** button.

■ **Figure 4.345**
The settings available within VRay for **DR** (***Distributed rendering***).

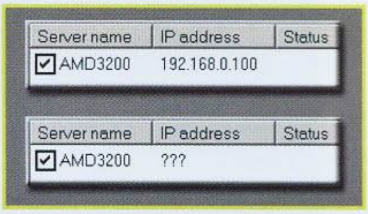


Add server - this button opens a new window where one can manually insert both the name of the PC in the Network and the IP address.

Remove server - deletes the selected PC server.

Find servers - (this function is disabled in this release).

Resolve servers - this button checks the availability of the PC server selected within the network. If the PC is irretrievable (network cable not connected, wrong network configuration, etc...) question marks will appear in the place of the IP.

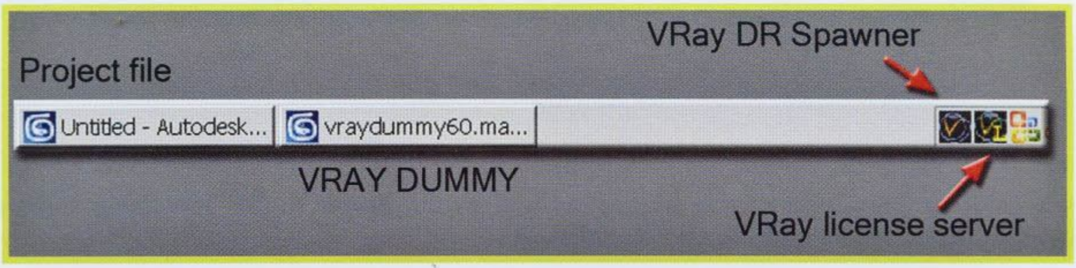


■ **Figure 4.346**
In the first case the ***DR*** recognizes the presence of a network server and proposes its IP. In the second case the PC is unavailable within the network and VRay is unable to retrieve its IP.

Using ***DR*** is very simple. First of all one must activate the ***VRay spawner***.

■ **Figure 4.347**
Activation of the ***DR spawner*** program.

This program activates a new 3ds Max session in server mode, by loading the file ***VRaydummy60.max*** into the 3D program and placing a new icon within the notification area.



■ **Figure 4.348**
Activation of a session of 3ds Max via the ***VRay spawner*** utility.

One must activate ***VRay spawner*** on all the PCs of the renderfarm. If this is not done, even if they are perfectly connected, ***VRay*** would not recognize them. It is also to be remembered that there must be a copy of 3ds Max and ***VRay*** installed on all PCs. ***VRay's DR*** allows one to render with a maximum of 10 PCs connected via network for each single license, independently from the number of cores/CPUs installed on each computer. One can also render via backburner by using, in our case, infinite nodes.

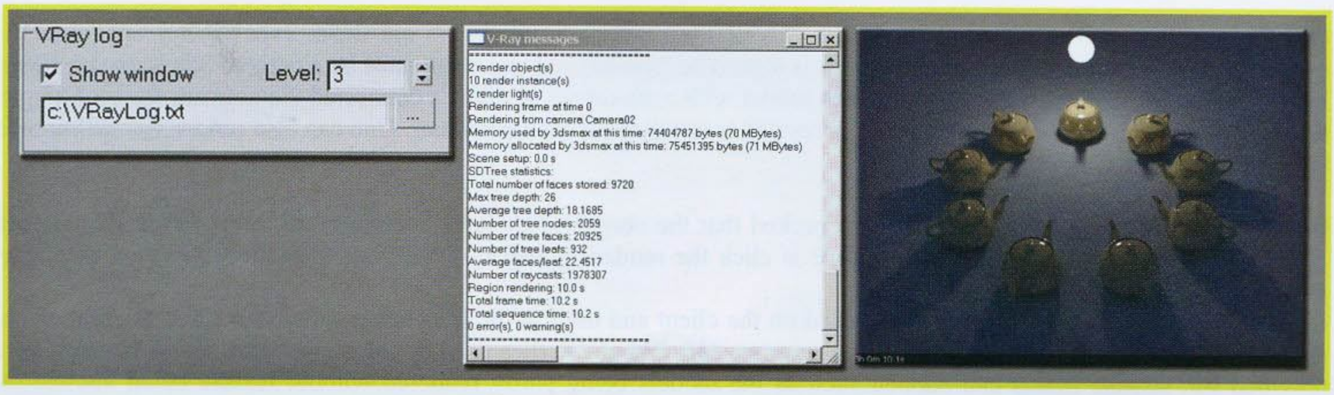
Once VRay spawner is active and one has checked that the network PCs have been inserted in the ***VRay Distributed rendering setting***, all that remains to be done is click the rendering button. ***VRay*** sees to the rest and distributes the ***buckets*** among the nodes.
One must also check that the plug-ins installed on the client and used in the files being calculated are also installed on the servers. Lastly, servers must have all the textures, ***IM***, ***PM*** or ***LC*** files used by the main client. If this has not been provided for, strange things can happen such as the models being partly rendered without texture being applied or renders missing the GI.

■ **Figure 4.349**
Problems due to missing textures on the PC server AMD3200. The grey objects are rendered without the textures being applied.

■ **Figure 4.350**
Problems due to the ***IM*** missing on the PC server AMD3200.

VRay log

When ***VRay*** calculates a render it activates a second window as well as the ***Frame buffer***, containing various information and statistics regarding the render itself, called ***VRay messages***.



■ **Figure 4.351**
The ***VRay log*** section with its corresponding window and a render of reference.

The information of this window is written in a file named ***VRaylog.txt***. By default the save path is located under ***C:***, although it is possible to change this via the appropriate button, represented by three dots. ***VRay messages*** allows one to read some of the information without having to manually open the file. The texts within ***VRay messages*** can be represented in four different colors:

.***RED*** ………. these are errors.
.***GREEN*** …….. these are warnings about possible problems.
.***BLACK*** …….. these are information messages.
.***ORANGE*** …… these are debug messages.

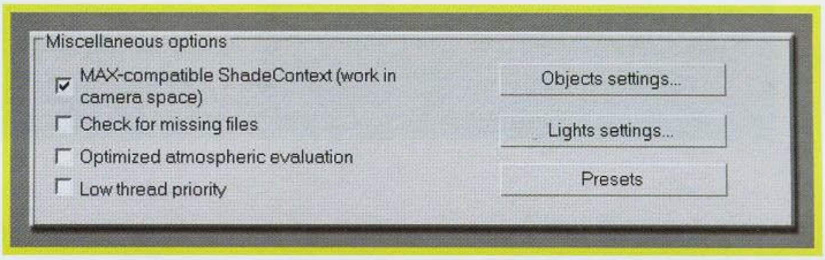
Log file - this parameter defines the location and name of the log file. The default path is ***C:\VRayLog.txt***.

Level - within ***VRay messages*** one can visualize 4 different types of messages, subdivided according to the amount of information made available:

1 - Only errors are shown.
2 - Both errors and warning messages are shown.
3 - Errors, warnings and information messages are shown.
4 - All messages are shown.

Miscellaneous options

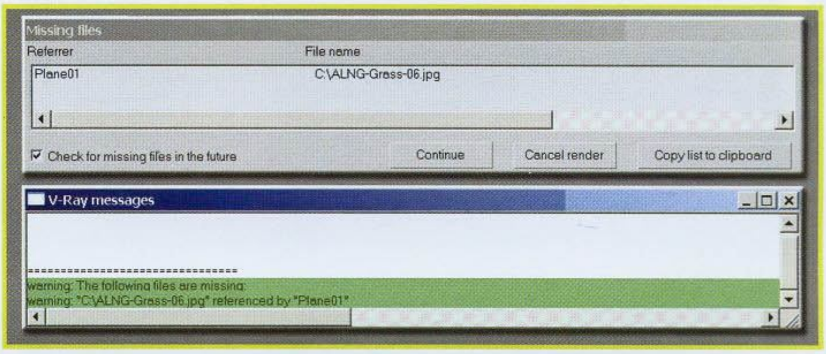
The parameters found in this section have purposes which differ greatly. It is difficult to categorize them in any of the rollouts we have seen so far and this is why a section has been created for them.



■ **Figure 4.352**
Parameters in the ***Miscellaneous options*** section.

MAX-compatible ShadeContect (working in camera space) - *VRay* carries out all its calculations in the global space. However some plug-ins for 3ds Max, especially atmospheric ones, are used in Camera Space because the Scanline engine supports these calculation modalities. In order to preserve compatibility with *VRay* too, the Chaosgroup renderer simulates this feature by converting its functionality. This happens for example with the Camera Correction modifier.

Check for missing files - if this is active, this parameter triggers the appearance of a dialogue window, just before render calculation, which indicates missing files in a session which has just been opened. Lost files are inserted into ***VRay messages***.



■ **Figure 4.353**
Menu and ***VRay messages*** notifying about missing textures needed for a file which has been opened. Notice the green color of the message which corresponds to a warning message.

Optimized atmospheric evaluation - usually in 3ds Max, atmospheric effects are calculated after the geometry has been rendered. This is sometimes useless if for instance this effect is very dense and opaque, enough to make the mesh invisible itself. By activating this parameter, *VRay* calculates the atmospheric apparatus first and after whatever lies behind it, providing the atmospheric effect is transparent enough to see what it hides. This is useful for example with ***VRaySphereFade***. This effect allows one to render only what the gizmo associated to it allows one to observe. Anything outside it is not computed, or rather, it is only computed if the ***Optimized atmospheric evaluation*** option is active. Otherwise, even if nothing appears in the final image, *VRay* calculates anything outside the gizmo in any case. This subject will be covered better later on in the book.

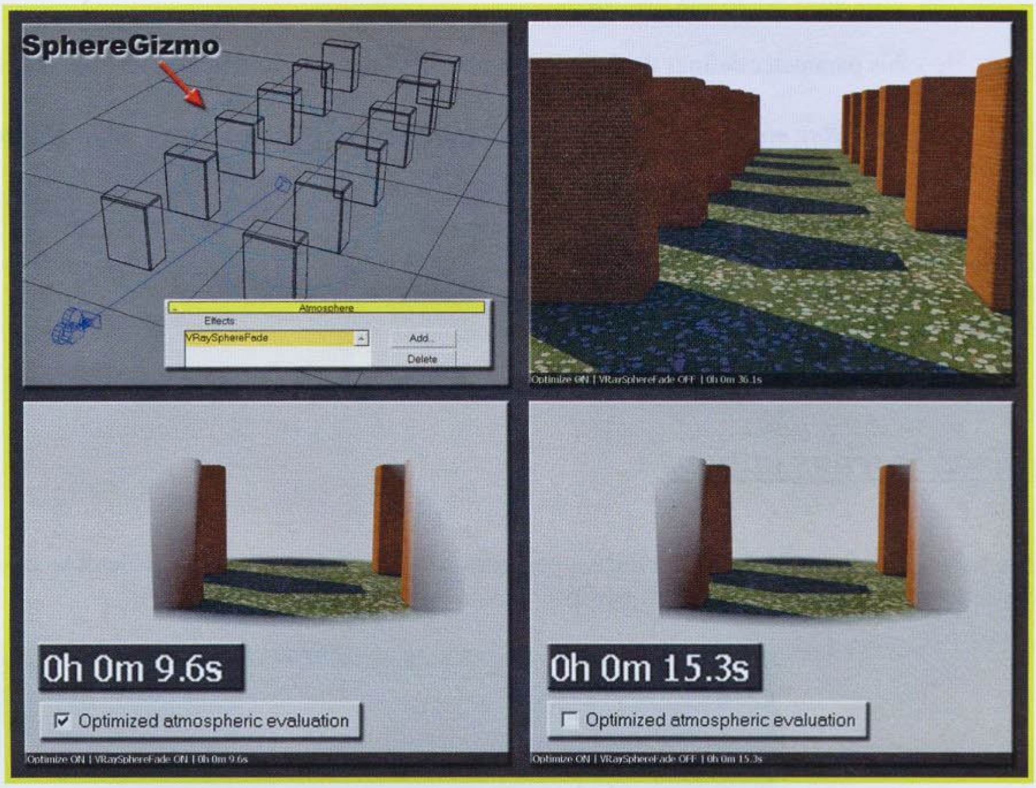


Figure 4.354 Reduction in rendering time thanks to the activation of the option ***Optimized atmospheric evaluation***.

Low thread priority - by activating this parameter, *VRay* sets the rendering to low priority so that one can comfortably work on the PC while rendering is taking place. *VRay* does not use all the computer's available resources, in particular of the CPU only for 3ds Max. When Windows requires power for another active program, *VRay* yields some of its CPU to this new process.

Object settings - this button activates a new window which has a complex look to it. This window allows one to set, for each geometrical object in the scene, some specific properties of *VRay*, as well as those found in the Properties panel of 3ds Max.

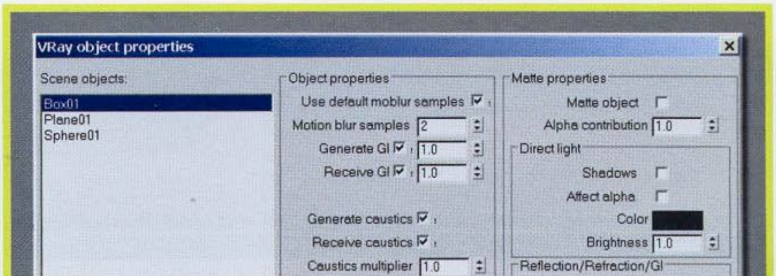
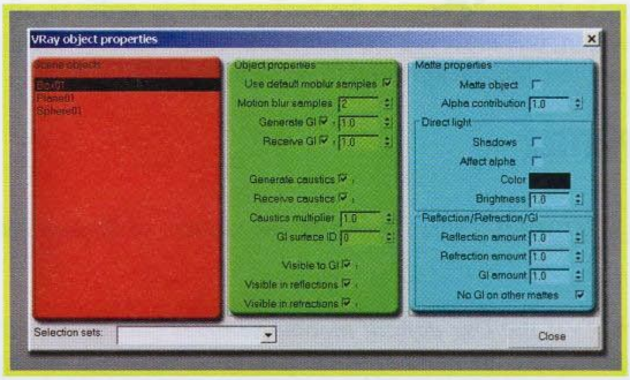


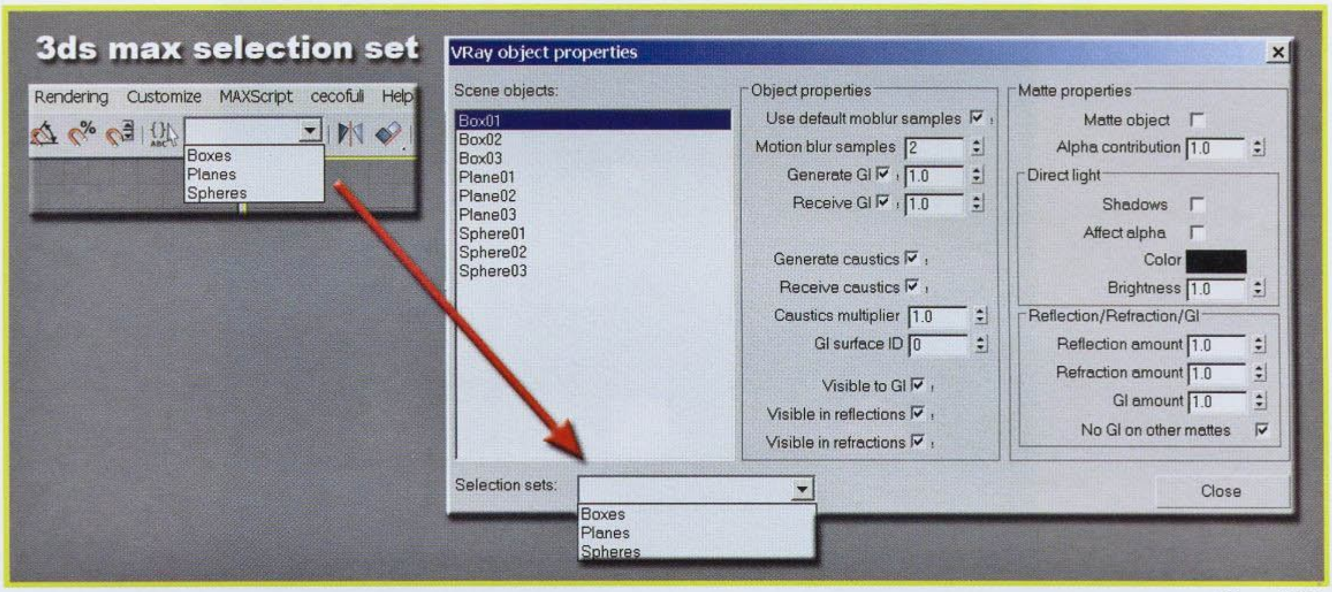
Figure 4.355 Thanks to the menu on the left one can simultaneously set the same properties for more than one object.

The window is subdivided into three areas:



■ **Figure 4.356**
The three areas which make up the ***VRay object properties*** window.

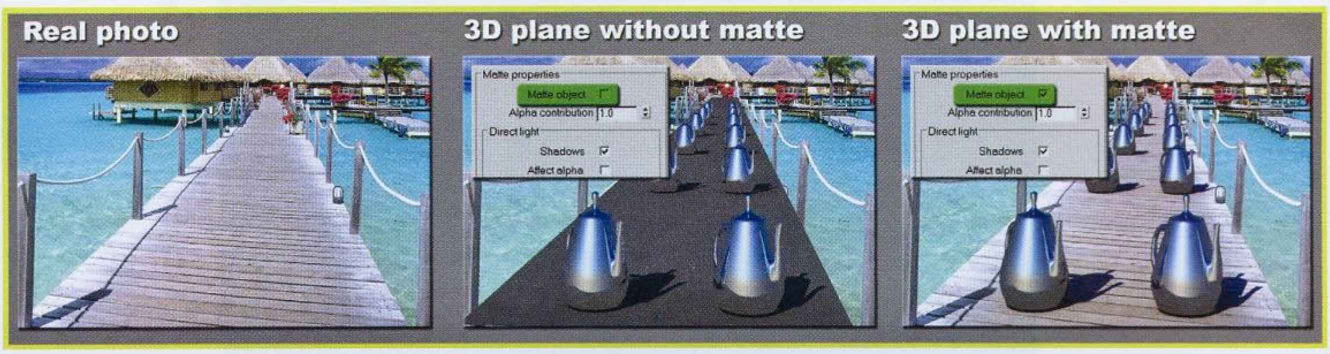
1. SCENE OBJECT: list of all the objects visible in the scene. If geometries are hidden or the layer assigned to them is deactivated, these items do not appear in the list. There is a drop-down menu called ***Selection set***, which allows one to recall selection groups created previously with the 3ds Max tool, ***Selection set***.



■ **Figure 4.357**
The ***Selection set*** function uses the 3ds Max selections.

2. OBJECT PROPERTIES: allows one to set important properties, such as ***Motion blur***, reception and generation of the Global Illumination or caustics.

3. MATTE PROPERTIES: a ***matte*** object is an object which is visible to rendering, but which has the ability to become "invisible", showing the ***Background*** behind it, as well as projecting and receiving shadows.



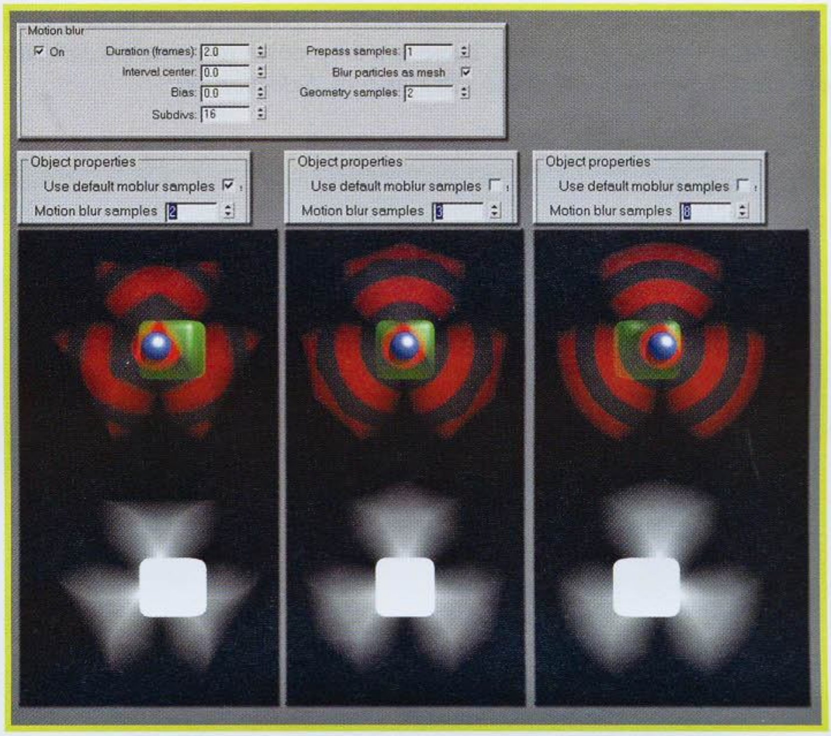
■ **Figure 4.358**
Example of use of the option ***Matte object***.

In the previous example a photograph was used as a ***Background***. After a plane and some geometrical objects were placed in such a way as to correspond to the background image. After having created a daytime light system, rendering was carried out. Obviously the plane covers the background. In order to allow both the background photo and the projected shadows to be seen, one must use the ***matte*** function. This way the plane becomes invisible, but able to make shadows on it visible and allow the ***Background*** to be seen.

VRay OBJECT PROPERTIES

Use default moblur samples - when activated, the number of ***Motion blur*** samples is globally defined by ***VRay: Camera*** rollout.

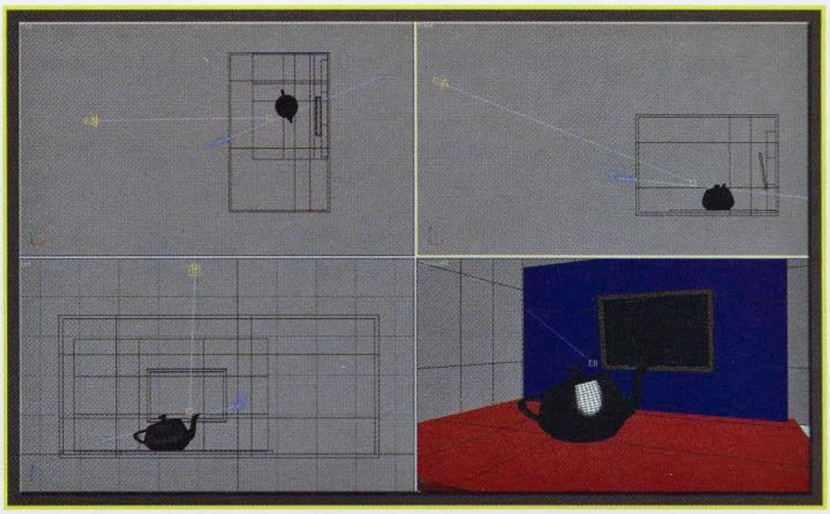
Motion blur samples - if the ***Use default moblur samples*** parameter is disabled, with this value one can specify ***Motion blur*** quality and therefore the number of samples to be used. For the explanation of this concept please visit the chapter regarding ***Motion blur***.



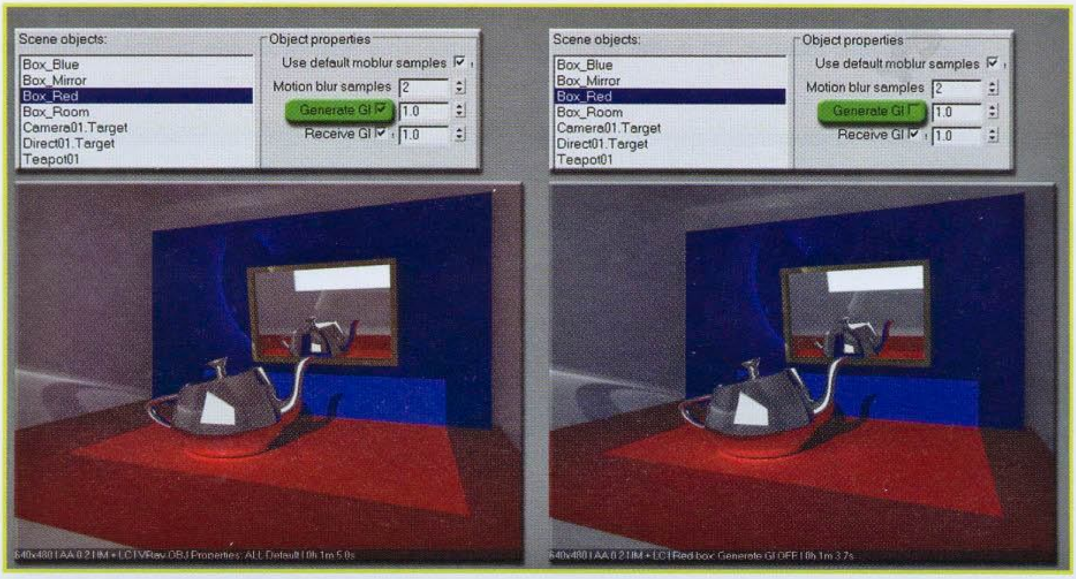
■ **Figure 4.359**
Three identical objects which are part of the same scene. Thanks to the additional properties made available by VRay, one can vary the number of samples for ***Motion blur***.

In the example, different sample values have been applied to the geometries. This way it has been possible to apply different sampling values to geometries in the same scene.

Generate GI - this option activates or deactivates object generation of indirect illumination. It is also possible to define the amount of indirect light generated or emanated.

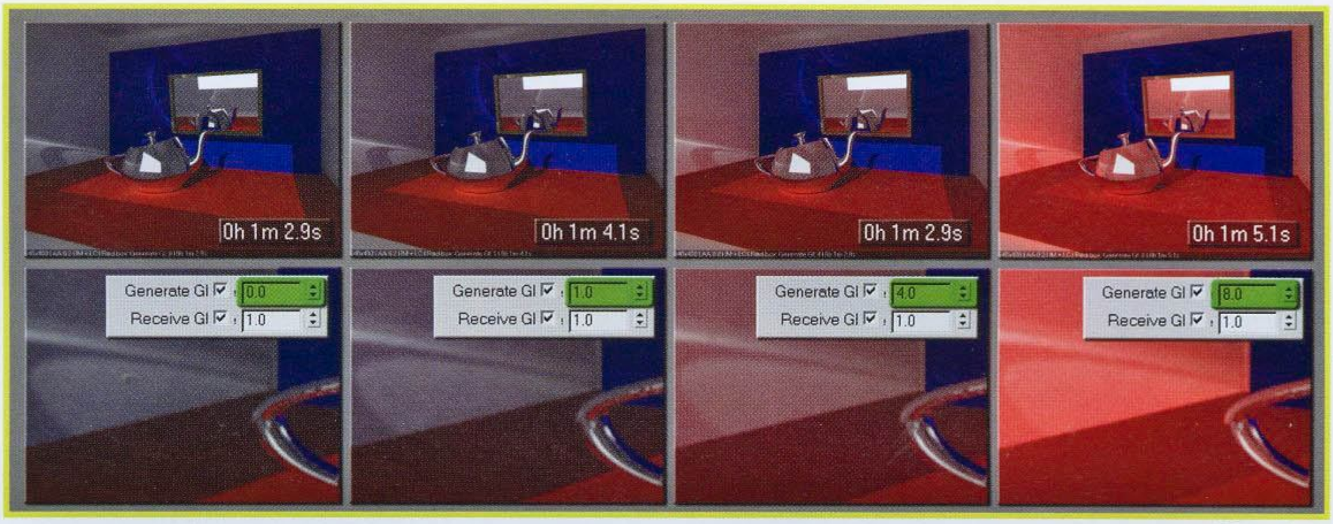


■ **Figure 4.360**
Scene used for the test. There are two light sources, ***skylight*** and a ***Direct Light***.



■ **Figure 4.361**
Deactivation of the ***Generate GI*** property of the Box Red object.

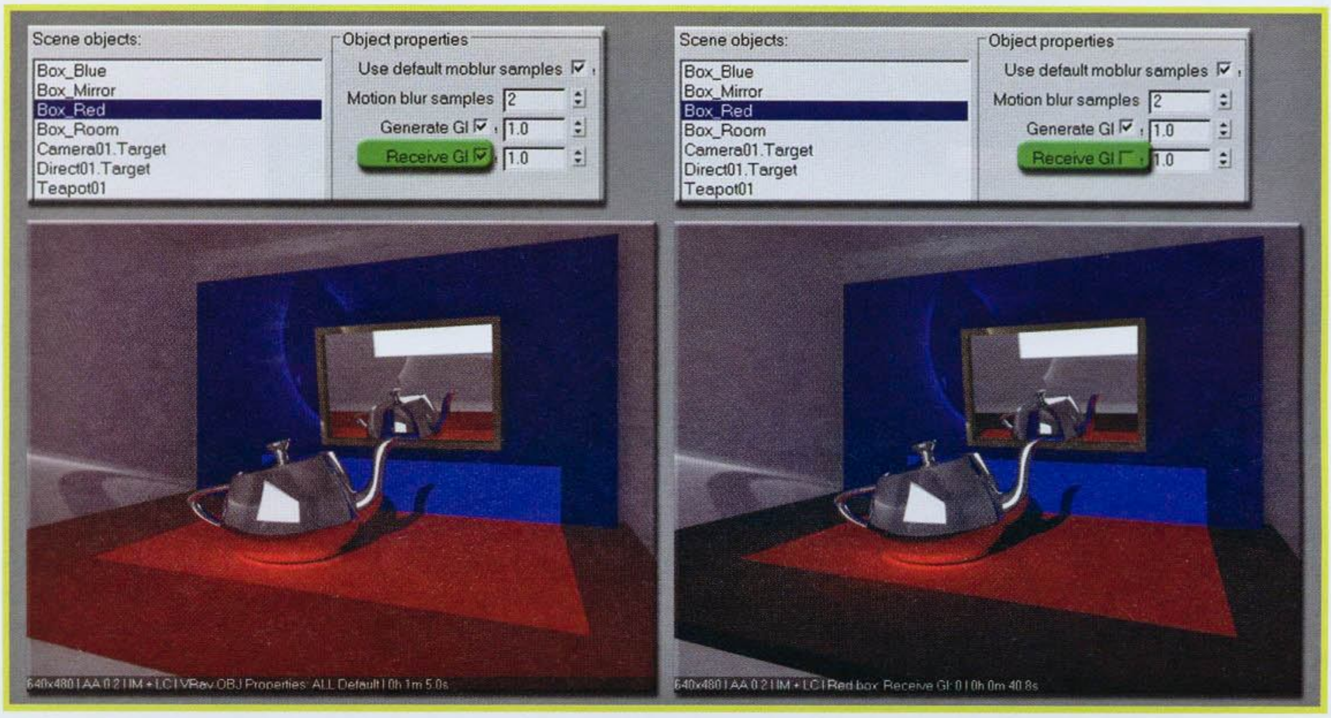
The effects of deactivation are clearly visible. The general red glow which fills the room, caused by the reflection of direct light on the horizontal plane, is missing completely when deactivating the ***Generate GI*** option, because this geometry no longer contributes toward the generation of Global Illumination. The room is well-lit in any case, thanks to white ***skylight***. One can notice a slight light-blue tone, due to the GI emitted by the horizontal wall. One can also define how much GI should be emitted by objects, by means of the ***Generate GI*** spinner.



■ **Figure 4.362**
Variation in the multiplier for GI generation.

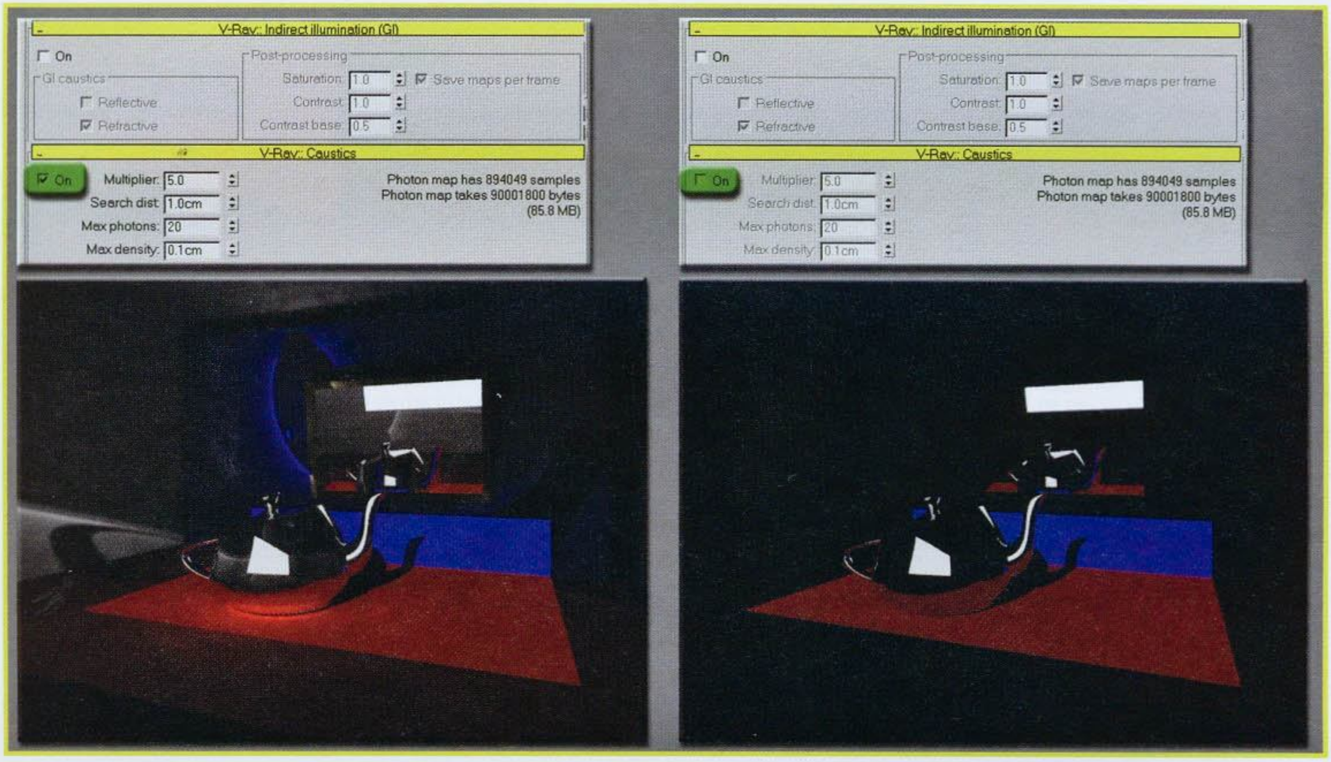
The result is an increase in the GI emitted by the red plane, with consequent saturation of the whole scene.

Receive GI - this option activates or deactivates reception of indirect light on the part of the object. One can also decide the amount of light the object should receive. As in the previous example, this parameter can be deactivated or modified with a spinner.

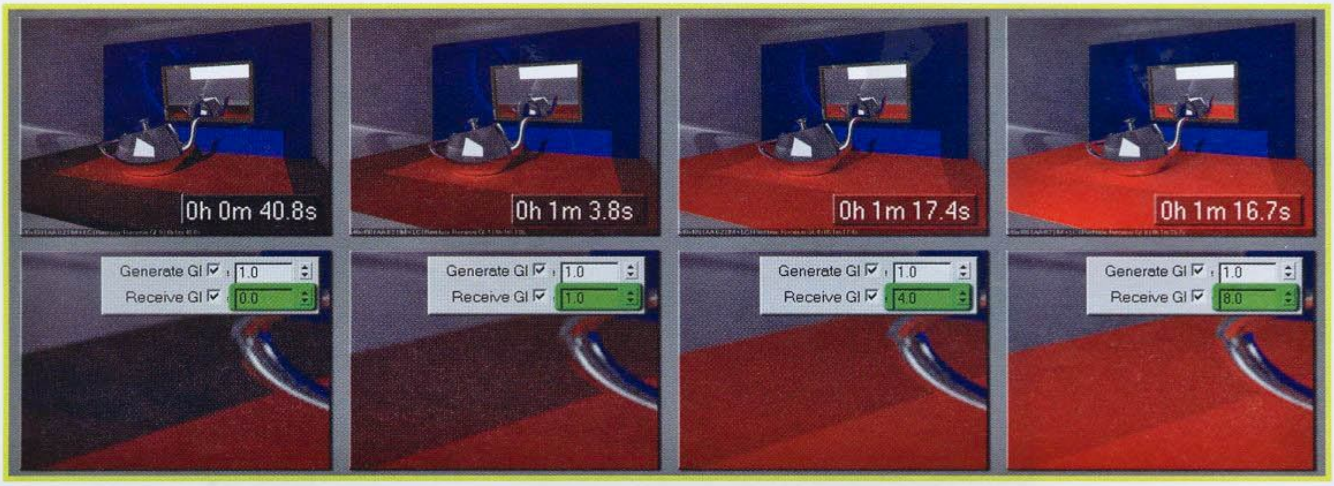


■ **Figure 4.363**
Deactivation of the ***Receive GI*** property of the object Box_Red.

In this case the red plane is unable to receive indirect illumination. It cannot therefore be illuminated by the indirect light coming from the Direct Light or the ***skylight***. The illumination, if it can be defined so, which allows one to observe some geometric portions not directly exposed to light is not produced by actual "light" but by the caustics generated by the teapot. If one observes the renders of the room without caustics and without GI, one can notice how this effect makes areas which are not directly exposed to light visible.



■ **Figure 4.364**
Activation and deactivation of caustics with GI deactivated.



■ **Figure 4.365**
Variation of the multiplier for the reception of the GI

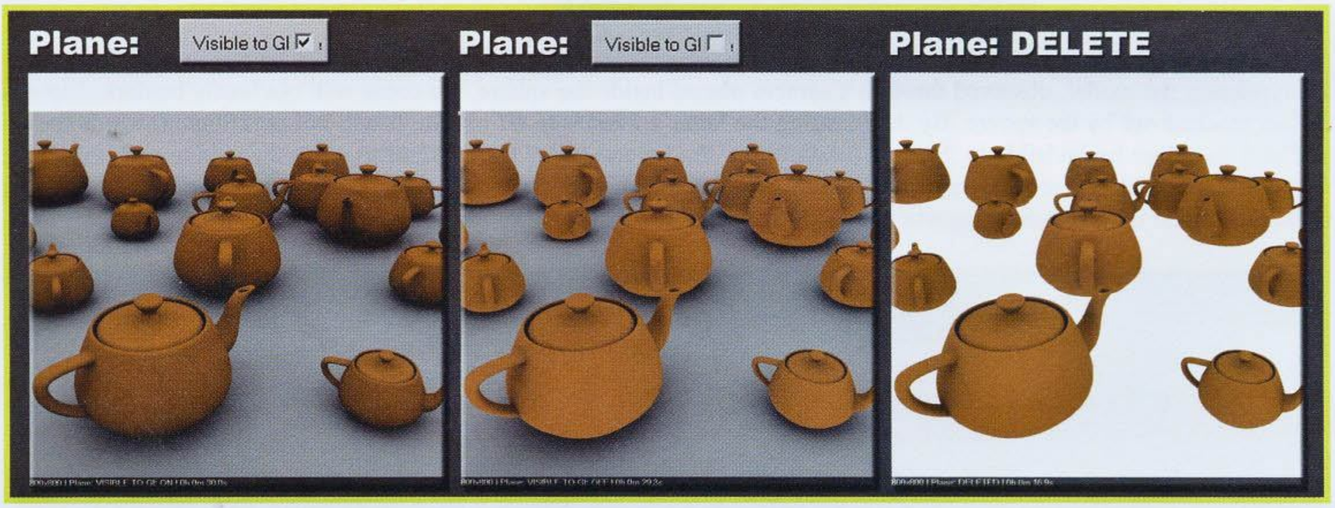
By modifying the ***Receive GI*** parameter the geometry receives a lot more GI than the normal amount, making it almost glow. The best advice is to leave these values set to one, as they are the ones which assure a physically correct representation of materials.

Generate caustics - this option activates or deactivates the generation of caustics on behalf of the selected object (see chapter on caustics).

Receive caustics - this option activates or deactivates the reception of caustics on behalf of the selected object (see chapter on caustics).

Caustics multipler - this multiplier amplifies the intensity of caustics produced by the selected object. This value affects the situation only if ***Generate caustics*** is active (see chapter on caustics).

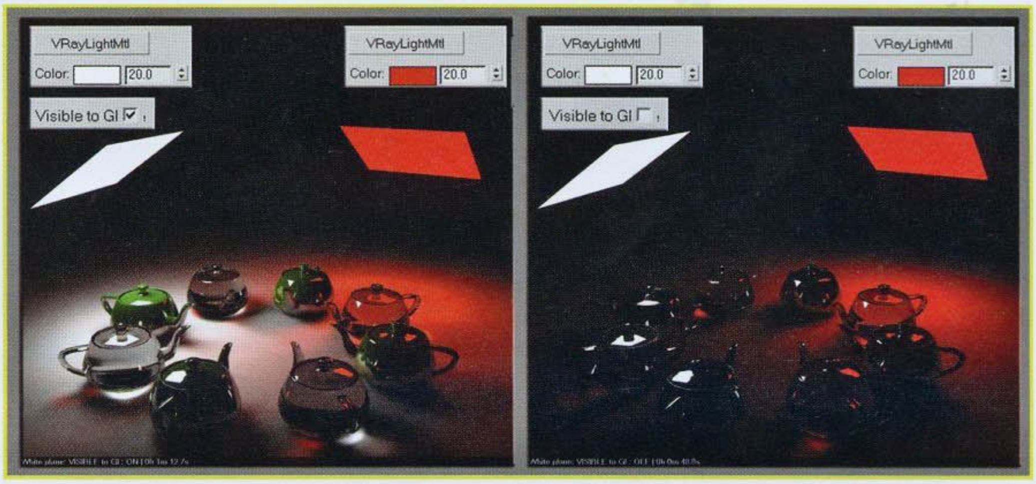
Visibile to GI - when active, this option enables the selected object to take part in the GI calculation. If it is turned off, the object in question does not participate towards GI calculation in any way. If the object is not visible to the GI, it will be able to receive it, but the rays of indirect light will not take its presence into account. In other words, for the GI this object does not exist.



■ **Figure 4.366**
Example of the use of the ***Visible to GI parameter***.

In this example some simple geometries are scattered on the plane. The scene is illuminated with ***skylight*** alone. In the first image everything is normal, instead in the second one the parameter ***Visible to GI*** of the horizontal plane is deactivated. The teapots undergo a change in luminosity, as if the plane they are lying on were not there. The plane instead is correctly illuminated.
By physically taking the plane away and re-rendering, the teapots are lit in the same way as in the second example. Basically the plane does not influence the GI of objects in the scene but is influenced by the GI in any case when the parameter ***Visible to GI*** is deactivated.

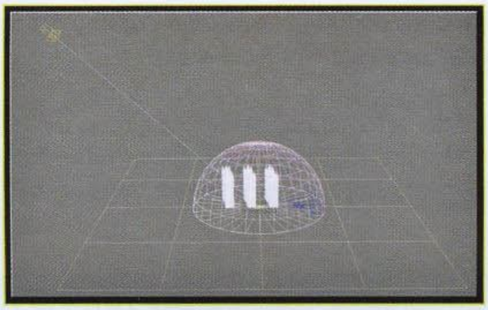
In this other example, two self-illuminating geometric planes project light on a few objects.



■ **Figure 4.367**
Other demonstrative scene of the use of the ***Visible to GI*** parameter.

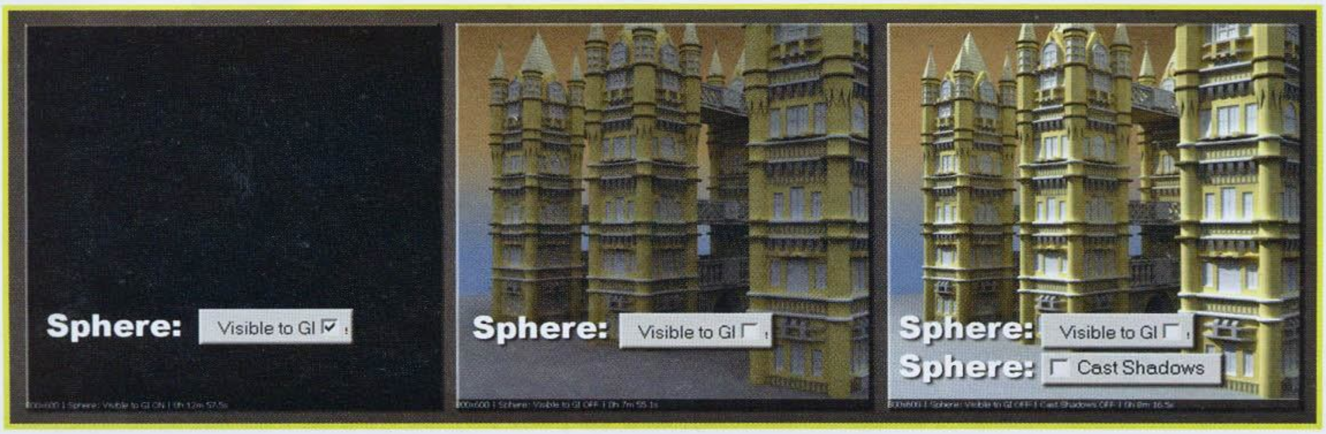
By deactivating the ***Visible to GI*** parameter of the white plane the light source no longer emits GI, although it still maintains its self-illuminating property.

In this last example, a building is placed inside a sphere and lies upon a plane. The sphere has inverted normals and outside it there is a Direct Light which, together with a ***skylight***, ensures a correct daytime light. The sphere has been created in such a way to be usable as an object which can be mapped, on which to apply a background such as a sky or landscape.



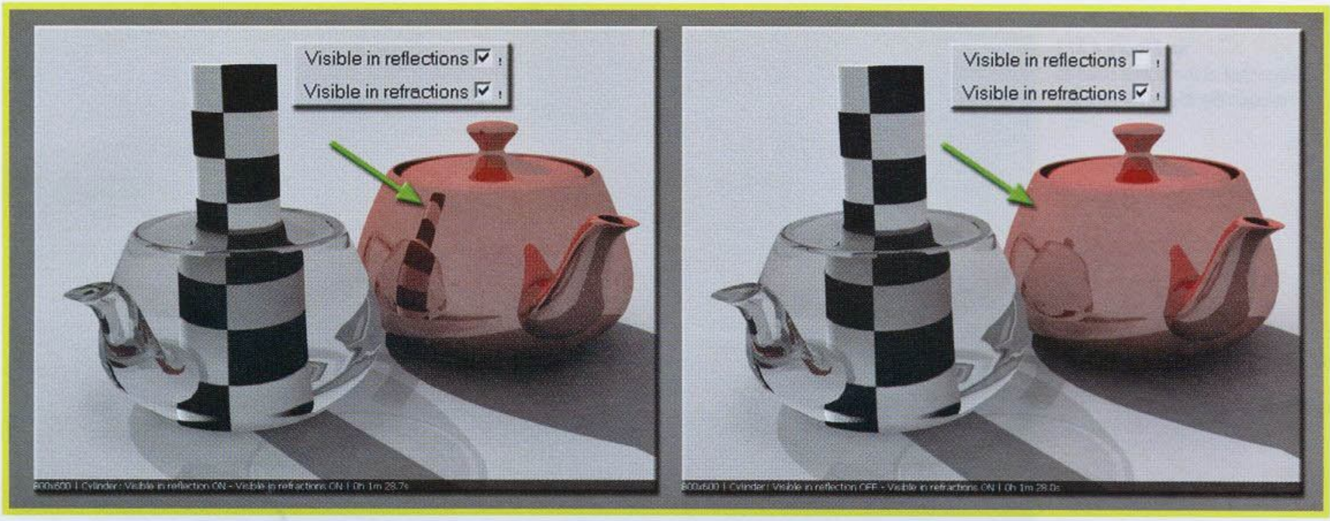
■ **Figure 4.368**
Screenshot of the scene.

By rendering the model, observed through a camera placed inside the sphere, the scene will obviously be dark. Light is in fact blocked out by the sphere. By deactivating the latter's ***Visible to GI*** option, it will not contribute towards the GI and will therefore be invisible to the rays produced by the GI and allow the ***skylight*** to cross the sphere and illuminate the building model. There is however still the problem of the Direct Light. This can be solved by deactivating the ***Cast Shadows*** option in the Object Properties window in 3ds Max.



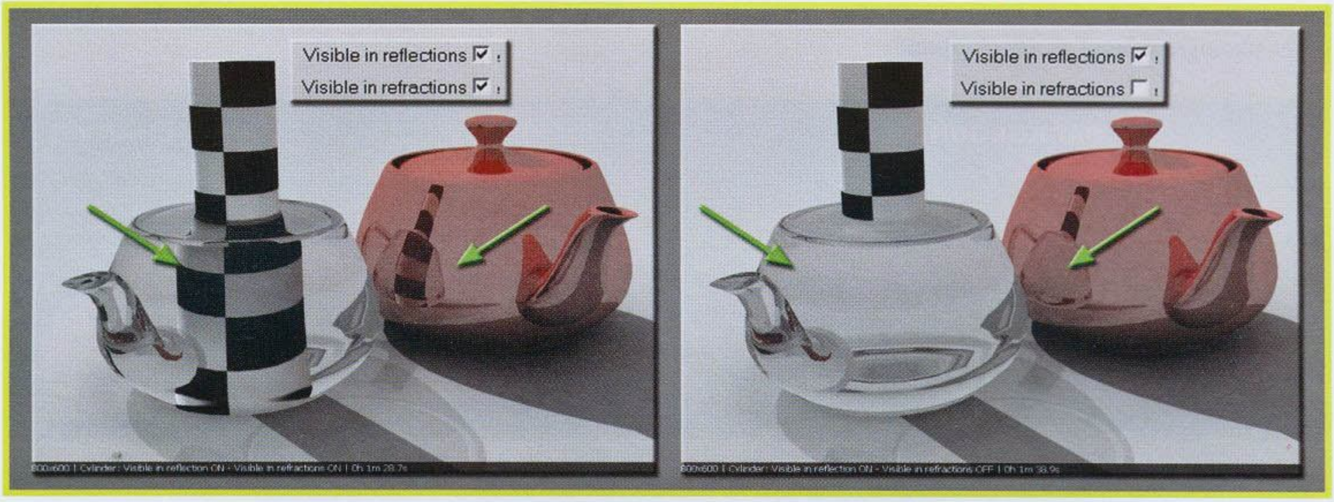
■ **Figure 4.369**
Practical example of use of the ***Visible to GI*** parameter.

Visibile in reflections - this option allows the activation or deactivation of the visualization of the object in all reflections present in the scene.



■ **Figure 4.370**
Deactivation of visibility for cylinder reflection.

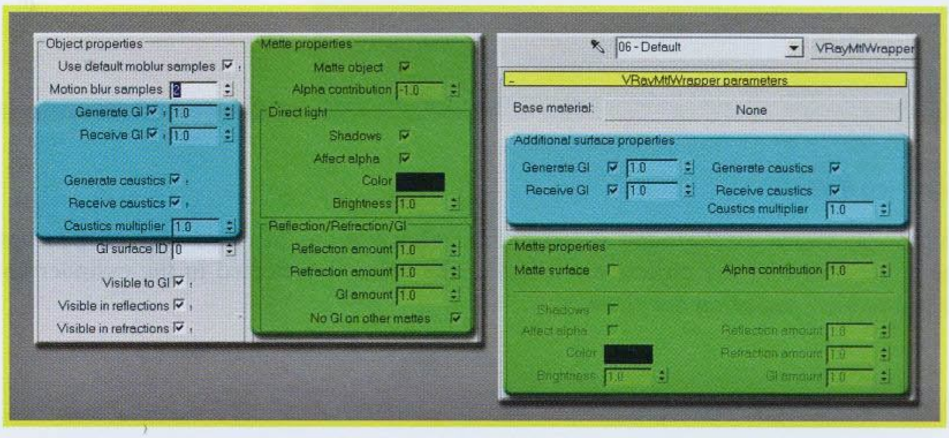
Visible in refractions - this option allows the activation or deactivation of the visualization of the object in all refractions present in the scene.



■ **Figure 4.371**
Deactivation of visibility for cylinder refraction.

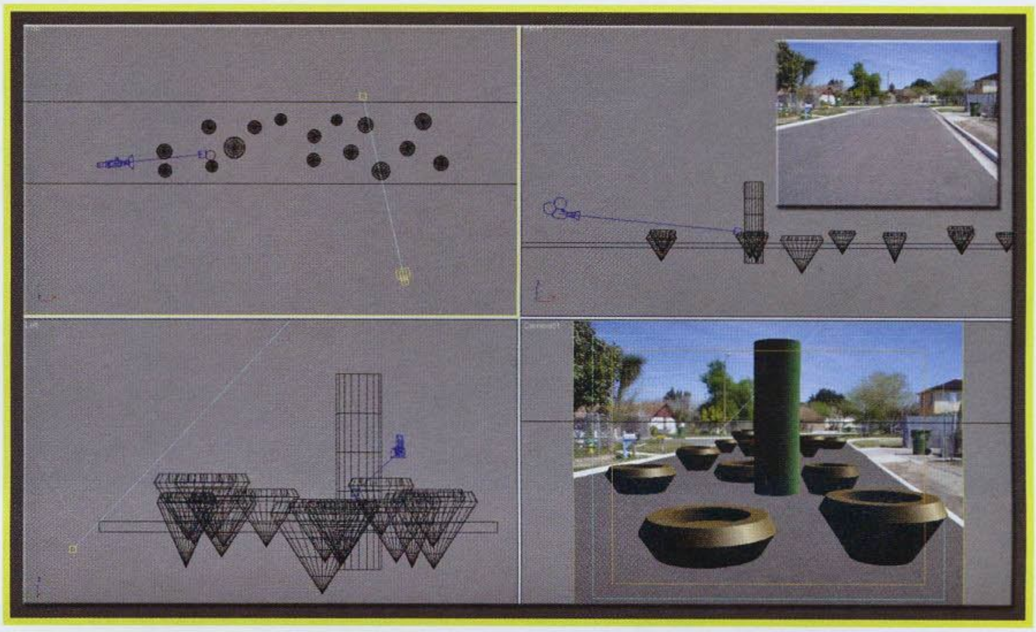
MATTE PROPERTIES

VRay does not support the Matte/Shadows shader of 3ds Max completely. This is why there are some options which allow one to replace this shader. It is possible to control ***matte*** functions both through parameters in the ***VRay object properties*** panel both via the ***VRayMtlWrapper*** shader.



■ **Figure 4.372**
Many options of the ***VRay object properties*** are to be found also in the ***VRayMtlWrapper*** shader.

Matte object - when activated, *VRay* transforms the geometry in a ***matte*** object. This object is not directly rendered within the render, but its place is taken by what is behind it. Usually this is the ***Background***. The plane however appears in reflections and refractions and generates indirect illumination, unless it is explicitly deactivated.



■ **Figure 4.373**
Screenshot of the scene used to explain the ***Matte*** parameter.

The scene we are examining is made up of a plane, long and narrow, on which many objects are placed. The whole thing lit by a ***skylight*** and a Direct Light. The camera has been placed so that these objects coincide with the perspective of the photograph used as a background.

■ **Figure 4.374**
Sequence: background, rendering and ***alpha*** channel in normal mode. No ***matte*** effect has been activated.

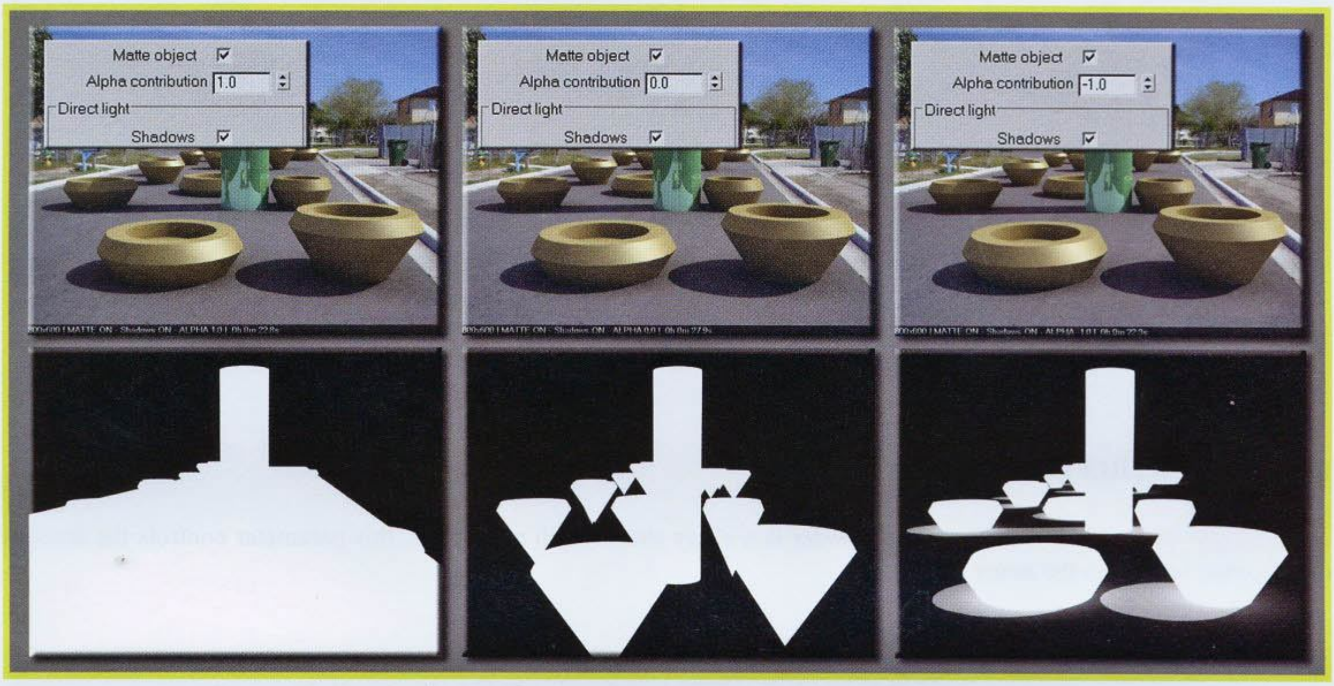
■ **Figure 4.375**
Same sequence with ***matte*** effect being applied.

Now the plane is invisible and allows Background visualization. Also portions of objects which penetrate it are not visible. However shadows generated by the objects are missing; the plane is reflected in the cylinder and the ***alpha*** channel contains all the geometries, including the plane.

In order to make shadows visible one must activate the ***Shadows*** option.

■ **Figure 4.376**
Activation of the ***Shadows*** parameter. Now shadows are visible, although the ***alpha*** channel remains unaltered.

Alpha contribution - this parameter allows one to control the way it is visualized in the ***alpha*** channel. Value 1.0 means that the object appears normally in the ***alpha*** channel. With value 0.0 the object is not computed in the ***alpha*** channel. Value -1.0 inverts the ***alpha*** channel of the selected object.



■ **Figure 4.377**
Activation of the parameter ***Alpha Contribution***.

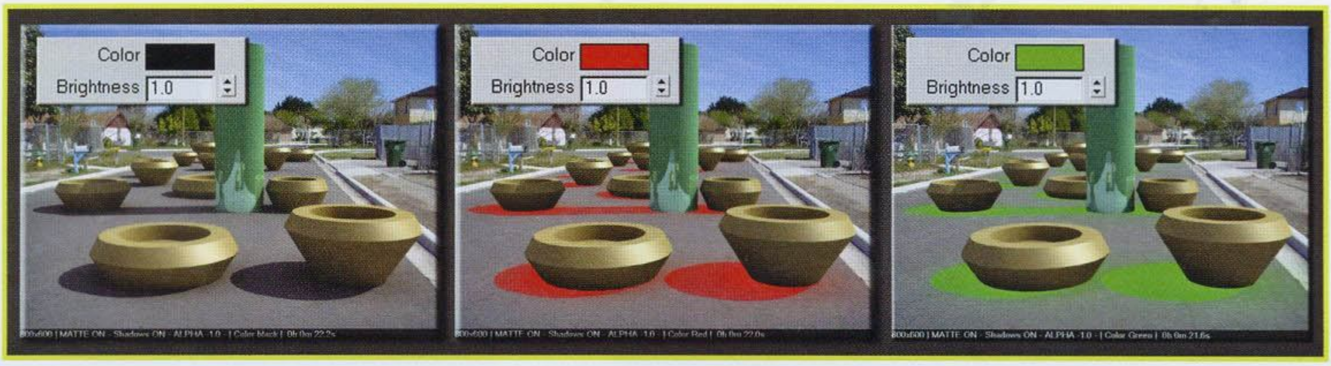
The parameter ***Alpha contribution*** modifies the ***alpha*** channel alone, whereas the RGB render remains unaltered. Now there are different types of ***alpha*** channel available, available for any demands of post-production.

DIRECT LIGHT

Shadows - by activating this option, the ***matte*** object can receive shadows generated by other objects projected onto it. In order to see an example of its use, observe the previous examples.

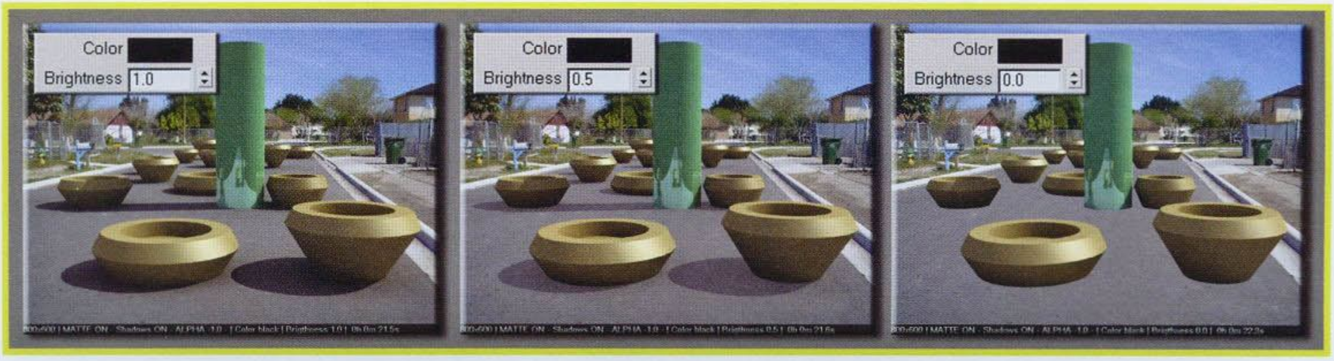
Affect alpha - the activation of this parameter enables the shadows to influence the ***alpha*** channel of the object.

Color - by changing the color inside the ***Color pick***, one can change the color of shadows of the objects.



■ **Figure 4.378**
Change in shadow colors via use of the ***Color*** option in the ***VRay object properties***.

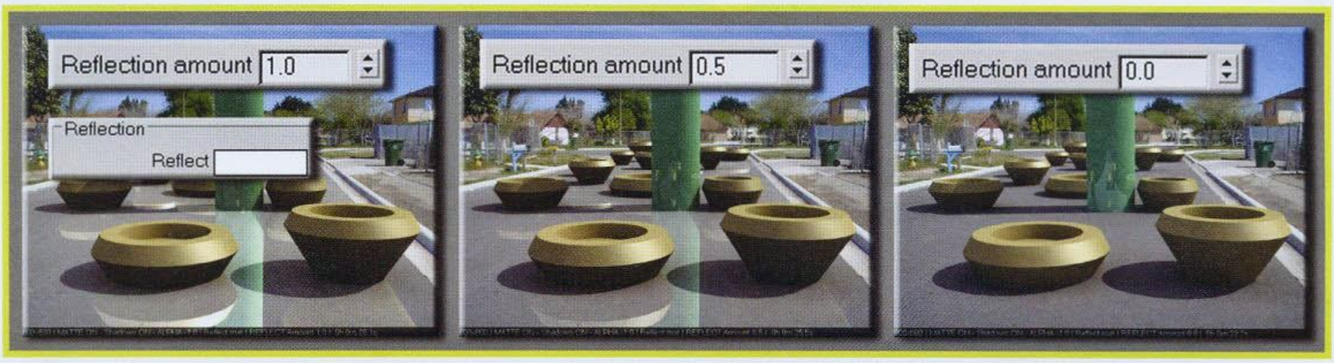
Brightness - with this parameter one can change the intensity of shadows, going from absent to perfectly visible. Consequently also the ***alpha*** channel is modified in order to adapt to the new intensity.



■ **Figure 4.379**
Change in shadow intensity by using the ***Brightness*** option in the ***VRay object properties*** window.

REFLECTION / REFRACTION / GI

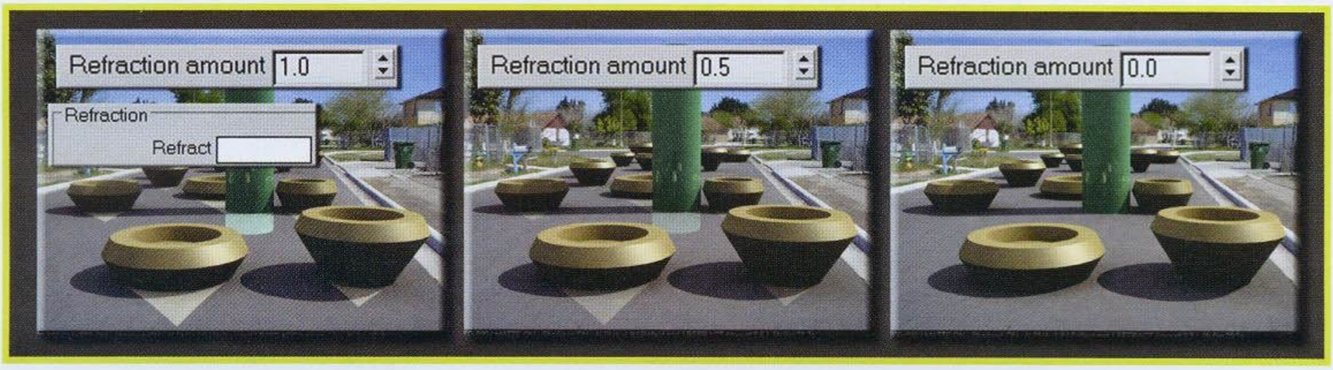
Reflection amount - if the geometry shader is a ***VRay*** shader with reflections, this parameter controls the amount of reflection visible on the ***matte*** object.



■ **Figure 4.380**
Change in intensity of the reflection on the ***Matte*** object. Notice how the tarmac goes from shiny to opaque.

Reflectivity has been added to the plane through the shader. Now the tarmac in the background photograph is reflective. With the ***Reflect amount*** parameter one can manage the amount of reflection.

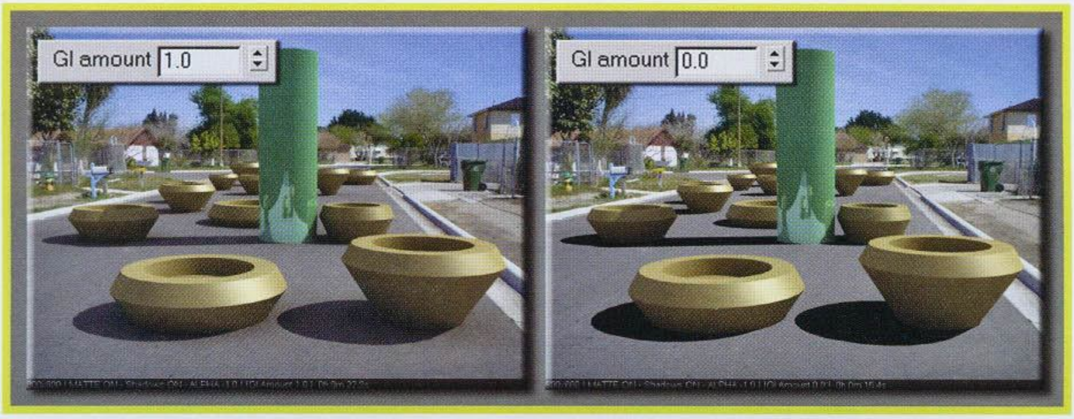
Refraction amount - this spinner, if the object's shader is a *VRay* shader with refractions, this parameter controls the amount of refraction visible on the ***matte*** object



■ **Figure 4.381**
Change in the ***Refraction amount*** parameter value.

In this case the road plane, modelled with a primitive box, has a certain thickness. By assigning refractivity to the base, the cones which intersect the plane are visible. With the ***Refraction amount*** parameter one can decide the amount of refraction of the ***matte*** object.

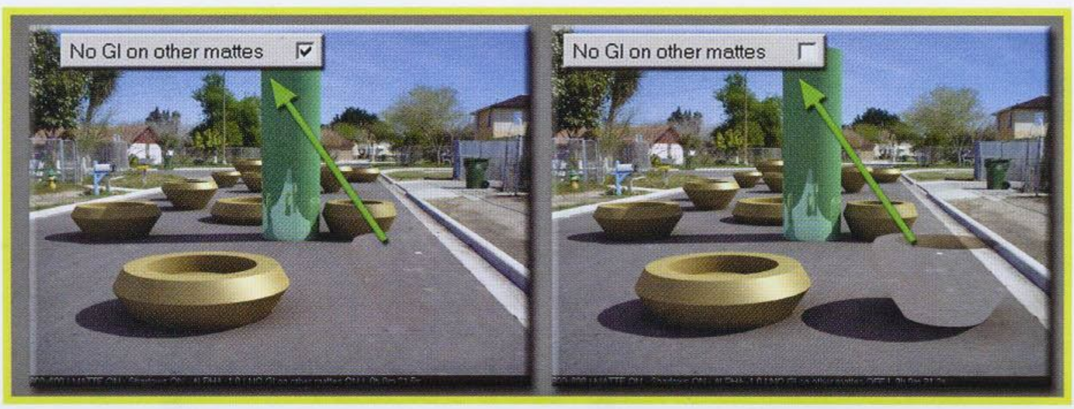
GI amount - this spinner controls the visibility of Global Illumination received by the object ***matte***.



■ **Figure 4.382**
Change in the ***GI amount*** parameter.

By resetting this parameter, GI is not calculated on the ***matte*** plane, leading to better rendering times. This means that shadows will be completely black.

No GI on other mattes - by disabling this parameter, the object appears as a ***matte*** object in reflections, refractions and GI of other ***matte*** geometries.

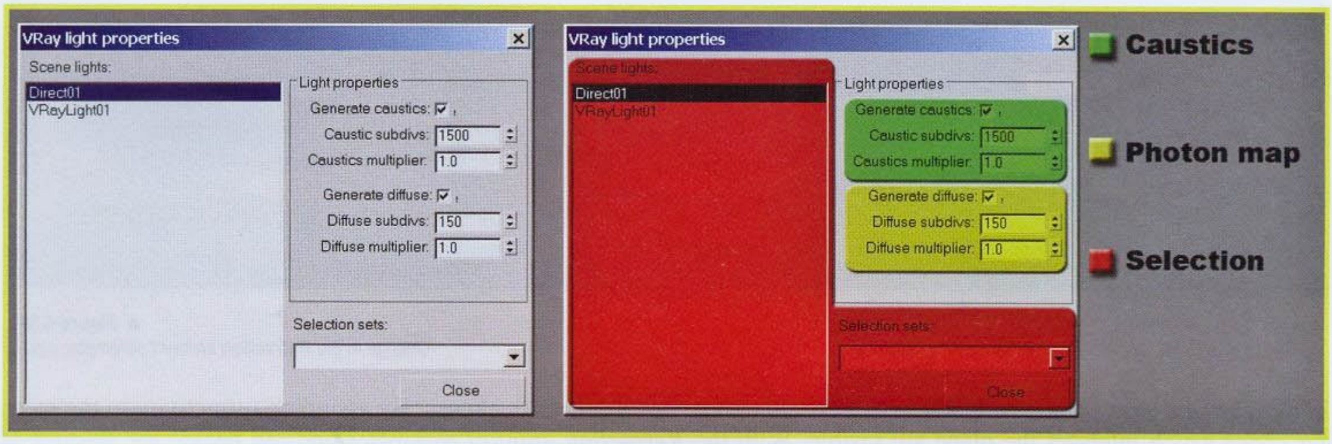


■ **Figure 4.383**
Results due to activation or deactivation of the ***No GI for other mattes*** option.

In the scene there are two ***matte*** options, the plane and one of the cones. This last one has had the ***No GI for other mattes*** option disabled. This way the GI and shadow generated by this object are visualized correctly both in reflections, as can be seen on the central cylinder, and also on other ***matte*** objects.

VRay LIGHT PROPERTIES

This section allows one to specify some properties of lights, both in *VRay* and also for 3ds Max. The parameters in this square involve caustics and Global Illumination.



■ **Figure 4.384**
VRay Light properties panel subdivided into three main areas.

All the parameters in this window have already been amply discussed in the section regarding caustics and the ***Photon Map***. For explanations on these subjects see the relative chapters.

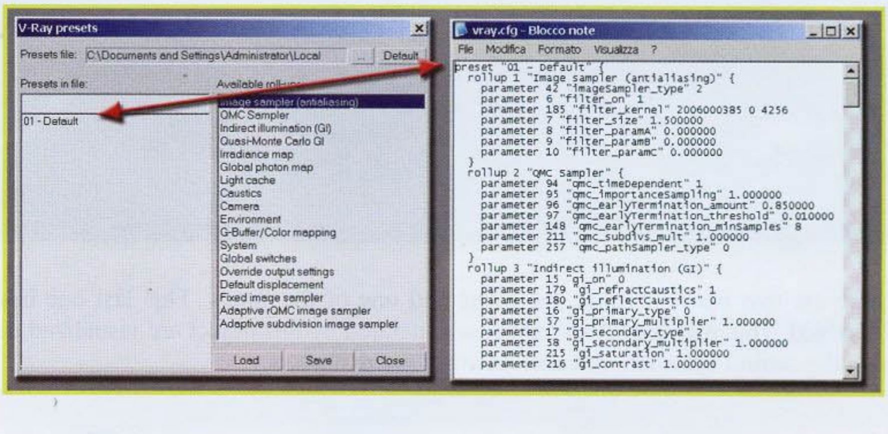
PRESET

By clicking on the ***Preset*** button a new window opens which allows one to save the settings of all values contained in the ***Renderer*** panel of *VRay*. So that they can be recalled and always available.

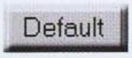
■ **Figure 4.385**
VRay Preset panel subdivided into its four main areas.

It is essentially made up of four areas. The first of these, emphasized in blue, represents the save path for the ***vray.cfg*** file which contains ***preset*** data. It is a simple text file and can be read with a notepad.

■ **Figure 4.386**
The ***preset*** file is a normal .txt file which contains all the information on setting parameters.

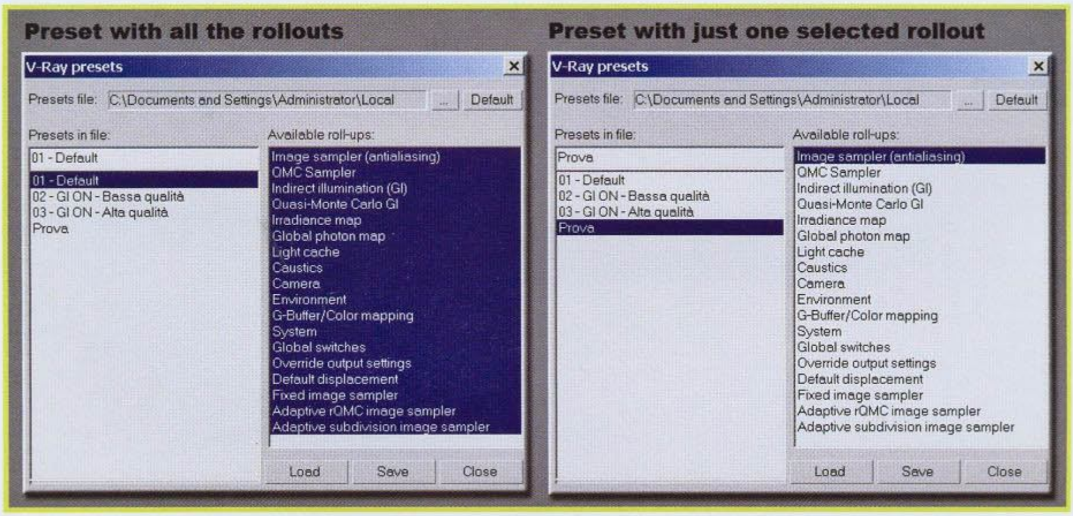


- it allows one to specify the save path of the ***VRay.cfg*** file.



- reset the save path for the ***VRay.cfg*** file to the default path.

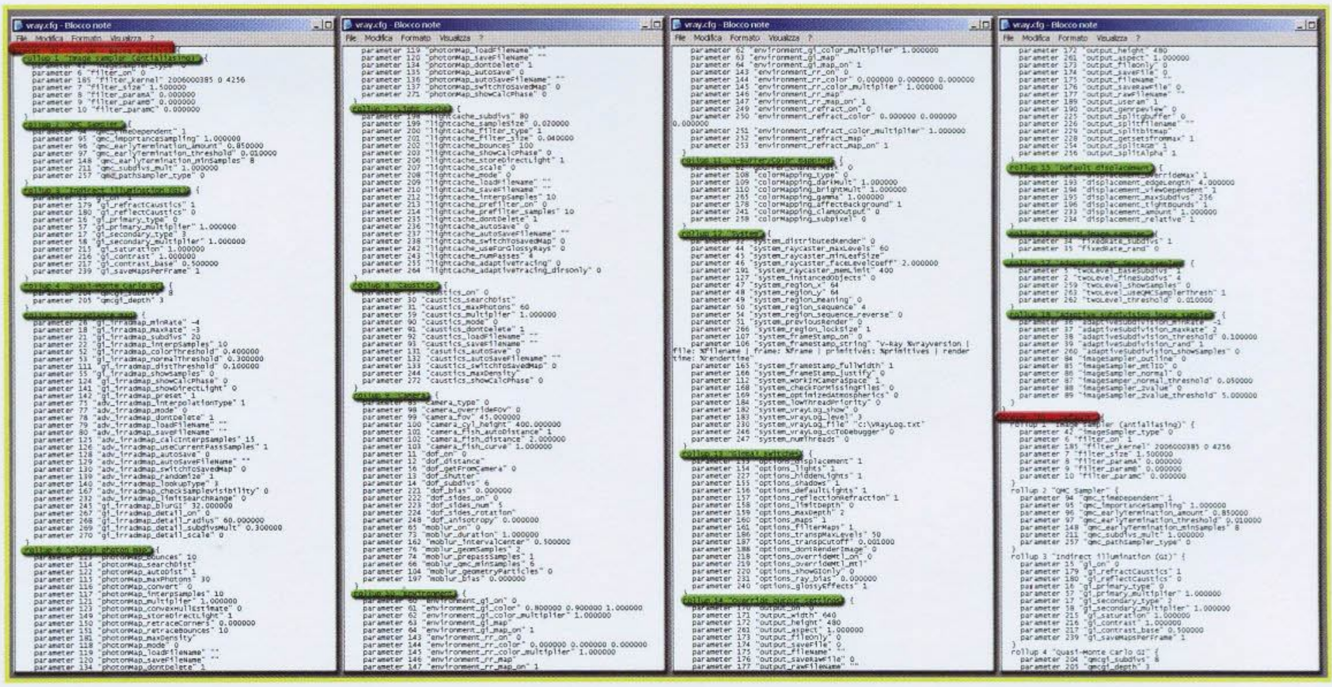
The section in green allows one to create a save ***preset*** with ***VRay*** settings. By writing the name of the preset in the appropriate space beneath the words ***Preset in file*** and clicking on the ***Save*** button, one can create a new status. It is also possible to create "partial" ***presets*** which only have a few settings. To do this one must select the rollouts of interest in the red section. ***Preset*** files are automatically ordered alphabetically.



■ **Figure 4.387**
For the preset "01 - Default" all the rollouts have been selected whereas for the trial presets only one.

In order to load status one must highlight ***presets*** and click the ***Load*** button. One cannot load partial rollouts. By loading a ***preset*** one uses all the parameters contained in it.

In order to change ***presets*** the only way is to manually open the ***vray.cfg*** file and change the parameters, which are easily understandable. The ***vray.cfg preset*** file is subdivided as a scheme.



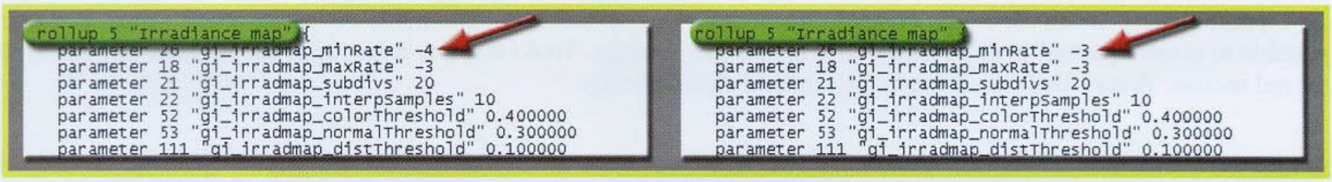
■ **Figure 4.388**
A part of the ***vray.cfg*** file opened with the Windows notepad.

The name of the ***preset*** is highlighted in red and is also modifiable. The 18 rollouts represented in green are included in the ***preset***. It is possible to manually modify all the parameters in order to update the ***preset***.

In this example the parameter ***Min rate*** of the ***Irradiance Map*** in the ***preset*** has been modified:

- 02 - GI ON - Low Quality.

It has been changed from -4 to -3. Now, by loading the ***preset*** again, the ***Min rate*** value will be equal to -3 and no longer -4.



```
rollup 5 "Irradiance map" {
    parameter 26 "gi_irradmap_minRate" -4
    parameter 18 "gi_irradmap_maxRate" -3
    parameter 21 "gi_irradmap_subdivs" 20
    parameter 22 "gi_irradmap_interpSamples" 10
    parameter 52 "gi_irradmap_colorThreshold" 0.400000
    parameter 53 "gi_irradmap_normalThreshold" 0.300000
    parameter 111 "gi_irradmap_distThreshold" 0.100000
```

```
rollup 5 "Irradiance map"
    parameter 26 "gi_irradmap_minRate" -3
    parameter 18 "gi_irradmap_maxRate" -3
    parameter 21 "gi_irradmap_subdivs" 20
    parameter 22 "gi_irradmap_interpSamples" 10
    parameter 52 "gi_irradmap_colorThreshold" 0.400000
    parameter 53 "gi_irradmap_normalThreshold" 0.300000
    parameter 111 "gi_irradmap_distThreshold" 0.100000
```

■ **Figure 4.389**
Manual change in ***preset*** save parameters.

CHAPTER 5: SHADERS AND MAPS

VRAY: SHADERS AND MAPS

One of the fundamental sectors of a rendering engine is definitely the shader department. In the past evolution of *VRay* ever more materials have been added. At the beginning, in version 1.0, only the *VRay* material existed (***VRayMaterial*** or ***VRayMtl***) and the correspondent *VRay* map. Today the choice is greater and so far 7 shaders and 8 maps are available, each with features which make them apt for certain situations.

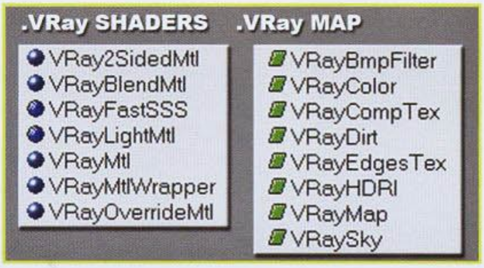


Figure 5.1
The shaders and maps available in VRay.

In the following chapters each shader and its map will be covered, with explanation of parameters and examples in images.
In order to load a shader into the Material Editor, one must click on the Standard button. Automatically a Browser will open which allows one to choose which shader should substitute the Standard one.

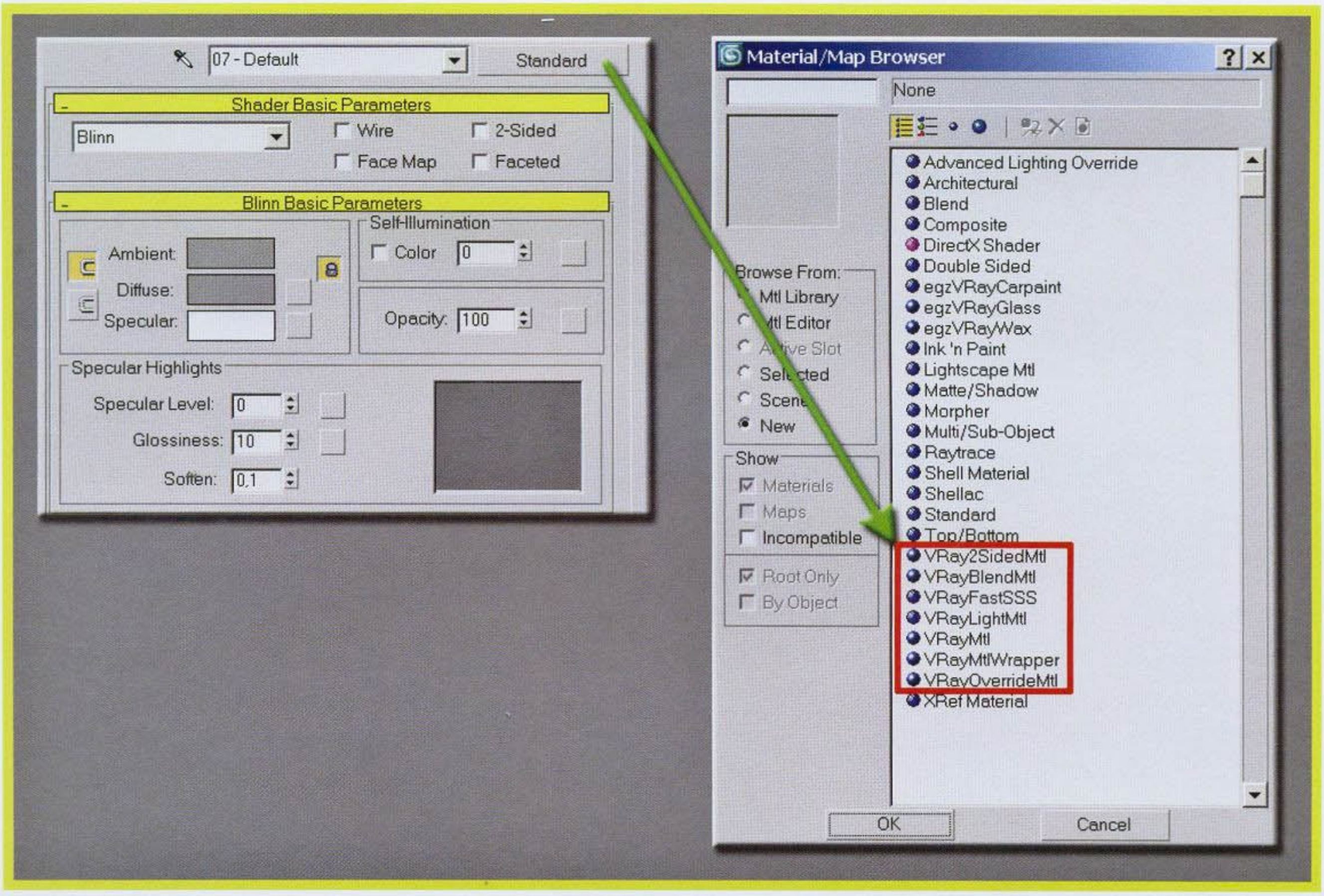
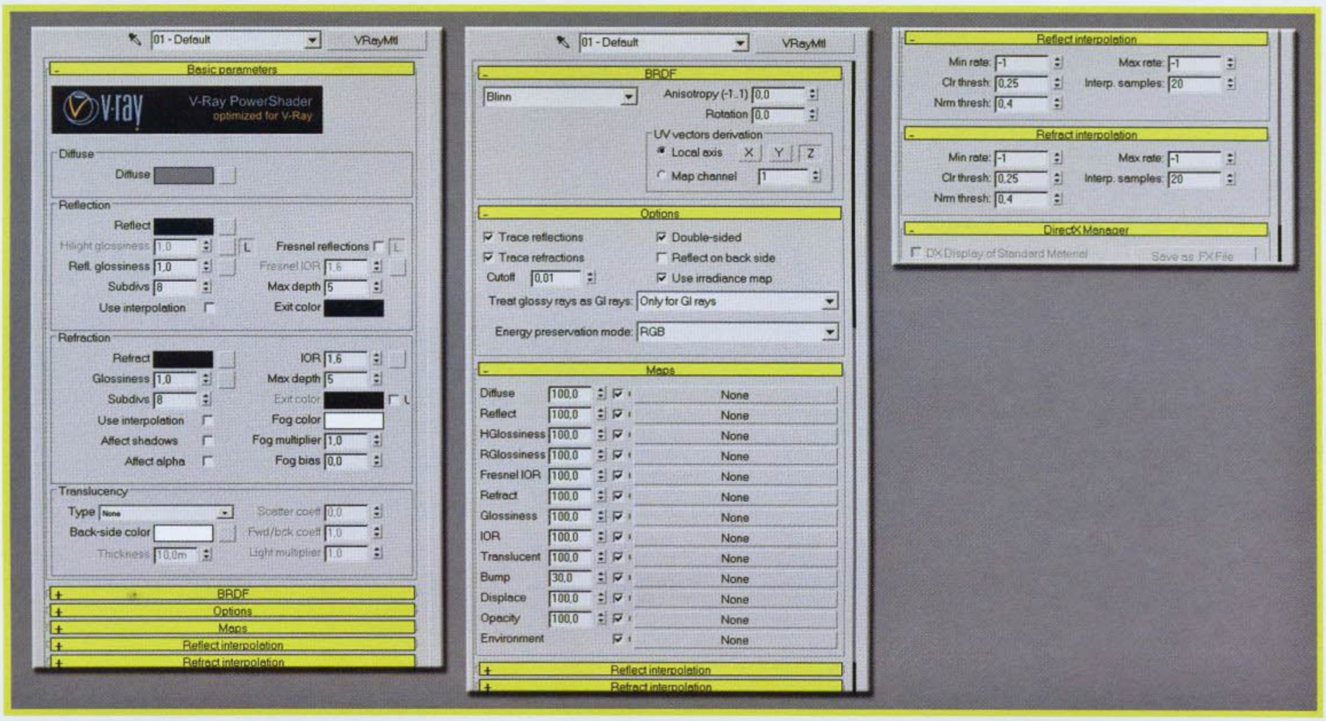


Figure 5.2
Replacement of the Standard material with a new one. All the VRay materials have been highlighted.

VRay Material : VRayMtl

INTRODUCTION

VRay, like many other graphic programs, allows one to render using the materials from 3ds Max (there are however some limitations, such as the impossibility of using raytrace materials or raytrace maps). This can be of help, for example, when one must render scenes created by others, which may contain common materials already applied. 3ds Max shaders, however, have certain limitations which can cause some problems, as we will see later on. Moreover, when using standard materials, one can encounter problems with illumination, rendering speed, etc... For this reason it is advisable, when possible, to use *VRay* materials and maps. This special material provides an improved distribution of luminous energy on the scene, faster renders, a better control of reflections and refractions, translucence, ***BRDF*** parameter management, ***Displacement*** and so on.



■ **Figure 5.3**
All the ***VRayMtl*** parameters.

The ***VRayMtl*** is subdivided into the following rollouts:

- Basic parameters.
- BRDF.
- Options.
- Maps.
- Reflect interpolation.
- Refract interpolation.

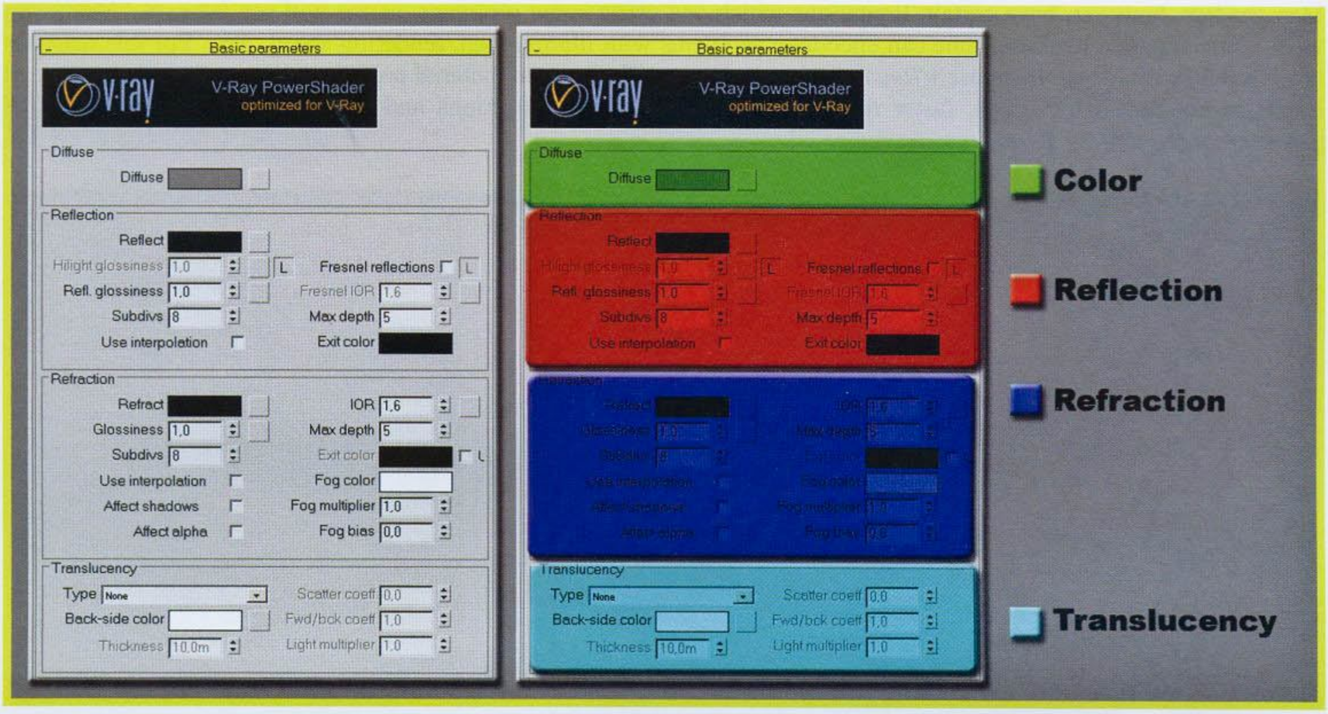
VRayMtl is a generic shader which allows one to create most materials. Plastic, glass, steel, skin, fabric, water, etc... can be reproduced with this shader. For loading and managing ***VRayMtl*** in the Material Editor one should read the 3ds Max online help.

PARAMETERS

The parameters for every single rollout will now be analyzed, starting from basic functions down to the interpolation of refractions.

Basic parameters

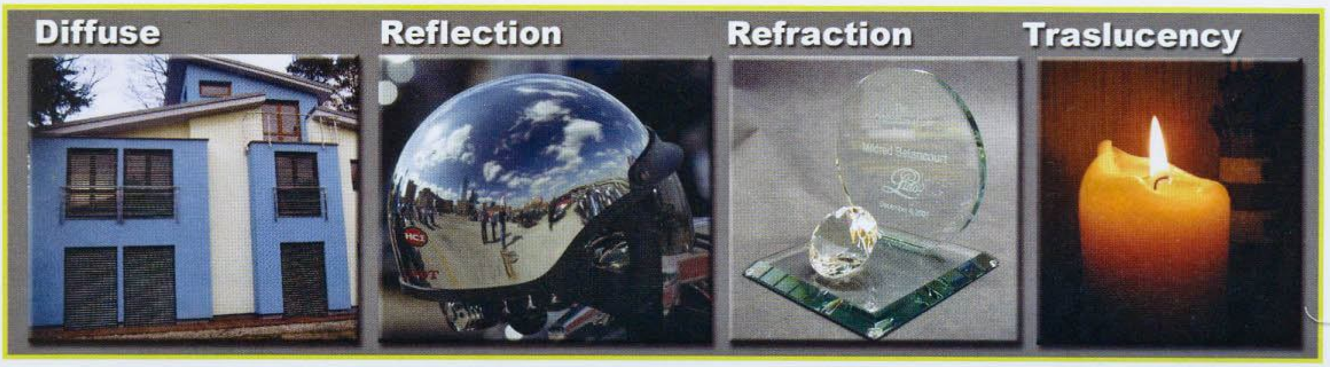
This rollout is the most complex of all the materials and maps in *VRay* and allows one to mange the main characteristics of an object. For this reason it is subdivided in four sections, with which it is possible to change any aspect of the material.



■ **Figure 5.4**
The four sections which make up the rollout for ***Basic Parameters***.

The four components are:

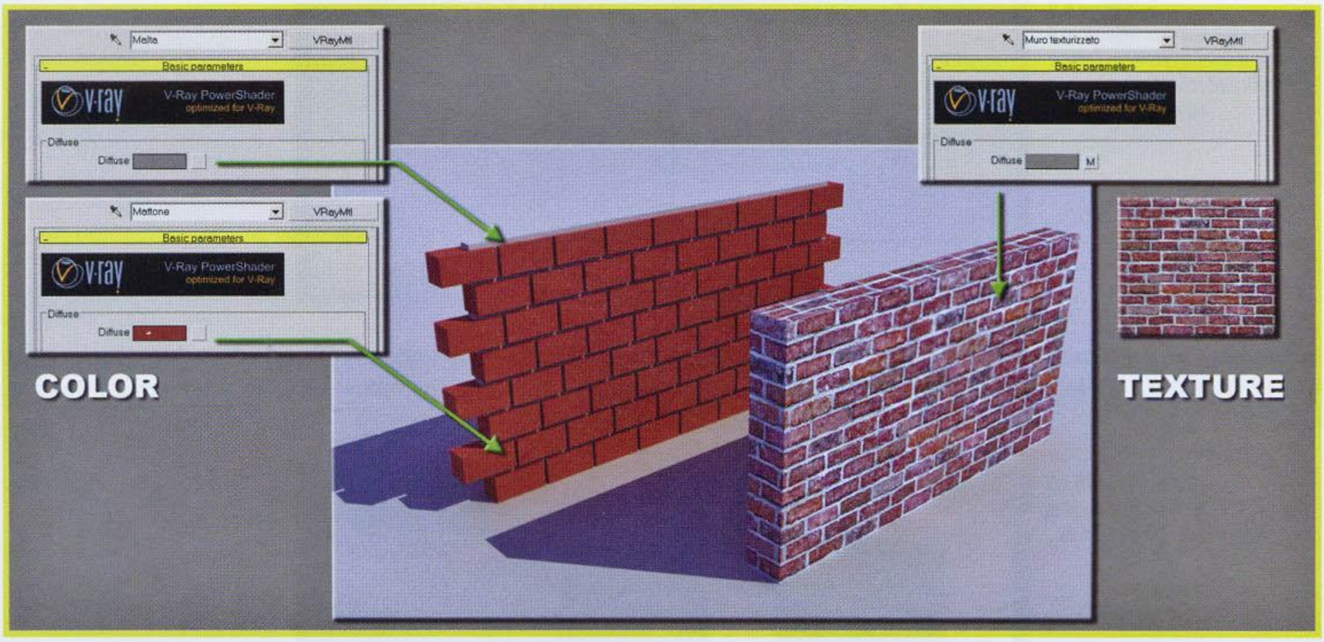
- ***Diffuse.***
- ***Reflection.***
- ***Refraction.***
- ***Translucency.***



■ **Figure 5.5**
Diffuse represents the color or the image used as a texture applied to the object. ***Reflection*** allows one to simulate typical effects of chrome materials. ***Refraction*** allows simulation of glass or transparent objects. ***Translucency*** simulates the transmission of light through the structure of an object.

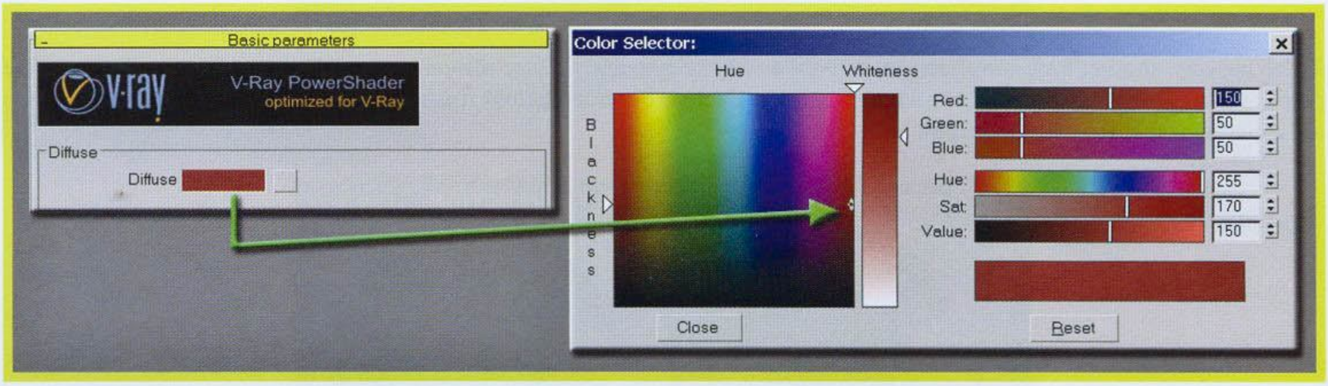
DIFFUSE

This term indicates the color component of the material. Each material has a color, be it black, white or grey, and the ***Diffuse*** channel is the fundamental characteristic of a material. If one wishes to apply an image in the place of a color, in this case we would call it a texture. For example, if one wants to simulate a wall of bricks, one could create each single brick and color them red, or create the whole wall and apply a map.



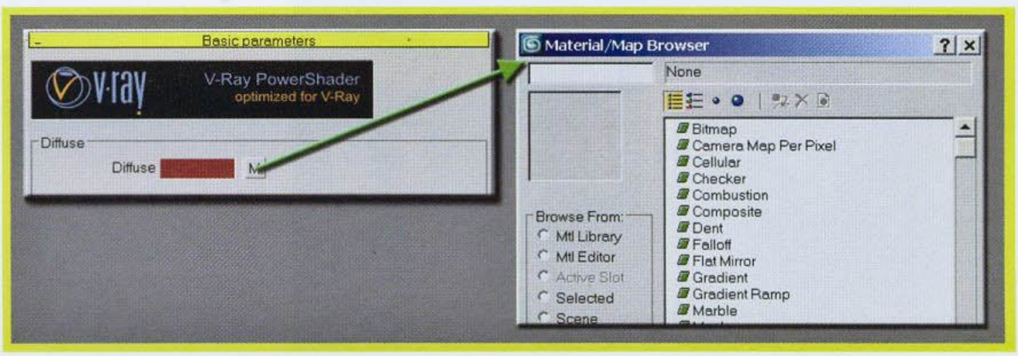
■ **Figure 5.6**
Different use of color and textures in the ***Diffuse*** channel.

In order to change the color, it is enough to click on the color-box. The ***Color selector*** will appear which is used for modifying RGB values, hue, saturation and contrast.



■ **Figure 5.7**
Changes being made in ***Diffuse*** color.

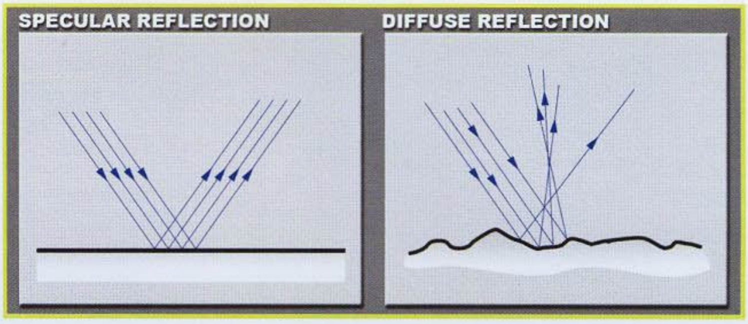
In order to load a texture, one must click on the square button next to ***Color switch***. The Browser then opens automatically and inside one can select the desired map, in this case a ***Bitmap***.



■ **Figure 5.8**
Application of a texture in the ***Diffuse*** channel. One can notice a capital letter ***M*** above the square button. This indicates the presence of a map in the ***Diffuse*** channel and that this is active and therefore visible during the rendering calculation. For other information on how to use 3dsMax maps, one should consult its online help.

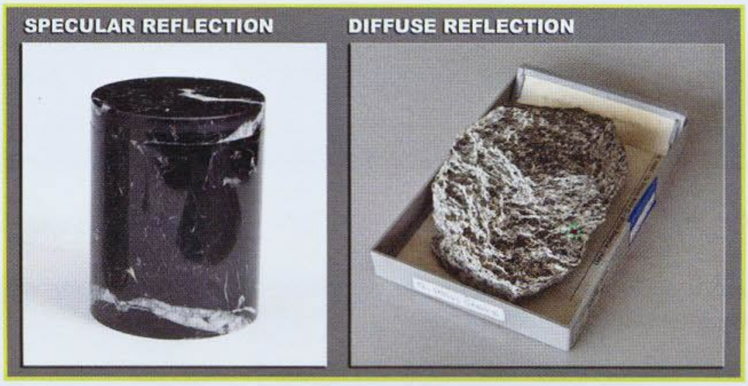
REFLECTION

Reflection and refraction are physical phenomena which take place when light interacts with materials. When energy radiates onto a body, some of it is absorbed, some is transmitted and some is reflected. Reflection, the subject of these pages, can happen in a specular fashion (regular or mirror reflection), that is, almost all in the same direction, or diffusely (diffused reflection) which means it reflects in different directions.



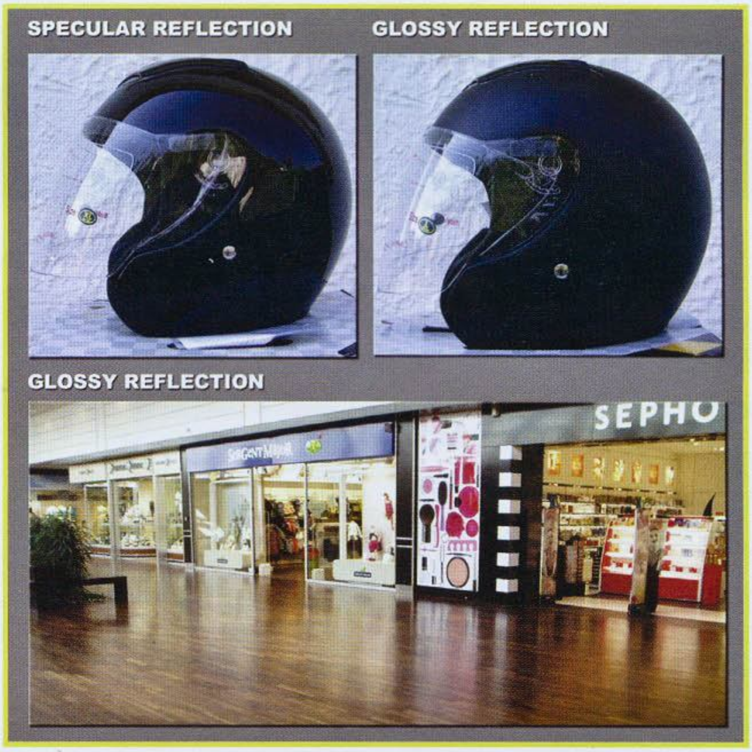
■ **Figure 5.9**
On the left a mirror reflection of light on a smooth surface. On the right diffused reflection of light on an irregular surface.

In nature these properties are the main difference between opaque and reflecting objects. Think of marble. One usually finds it in houses or churches and it is extremely shiny, but in its natural form it is opaque and does not reflect much at all.



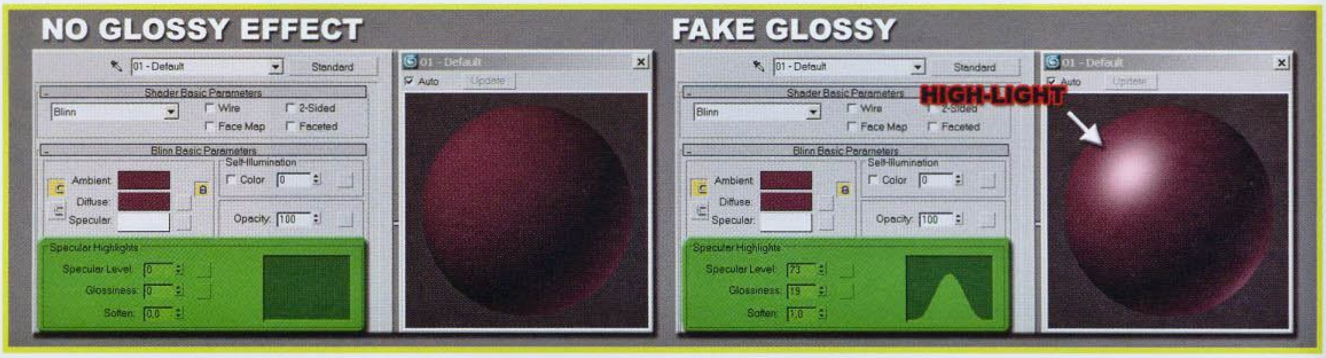
■ **Figure 5.10**
Examples of mirror reflection and diffused reflection in nature.

There is lastly another type of reflection, as well as mirror-like and diffuse reflections, which is extremely blurry, to the point that one cannot even perceive what is being reflected, but nonetheless it is a form of reflection. These reflections are called ***glossy*** reflections in jargon.



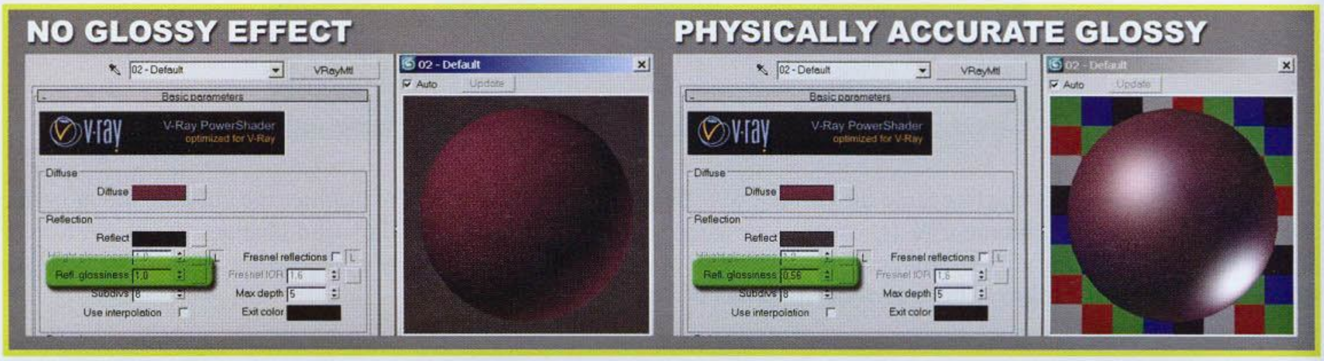
■ **Figure 5.11**
Examples of mirror and ***glossy*** reflections.

On the right-hand helmet we can observe mild tones of blue, due to the reflection of the sky. The same phenomenon happens on the pavement of the mall, where the shop windows are reflected on the wood but are also blurrier the higher they get.
A product of this behavior are so-called ***Highlights***, literally "high-up lights". In 3ds Max one can use the Standard material to create blurry reflections or ***Glossy reflections*** which are not physically correct, in other words "false" ***Highlights***. In order to do this one must change parameters within the section called ***Highlights***.



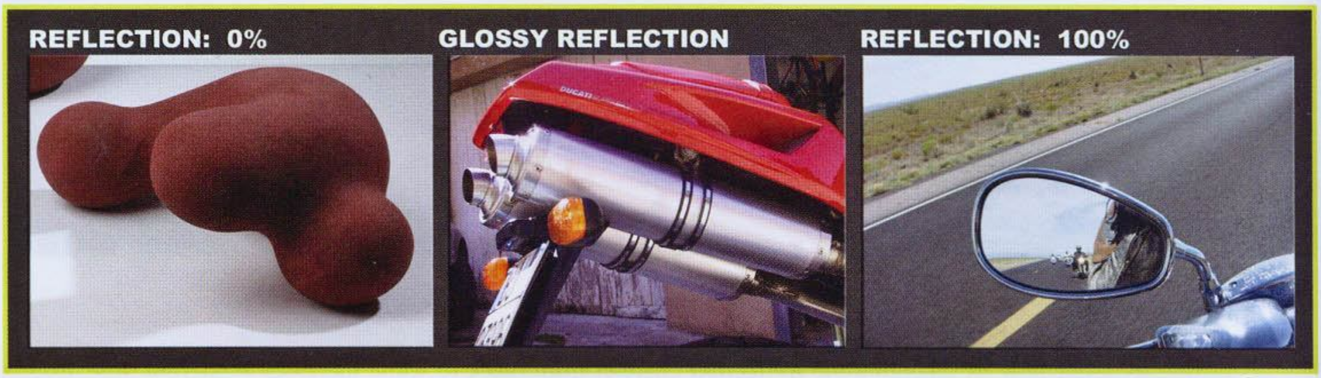
■ **Figure 5.12**
Simulation of blurry reflections with a Standard material.

In this case the simulation of blurry reflections creates an improvement in the shader, by giving the material a certain "movement". This trick can be replicated also within the ***VRayMtl*** and is useful in those cases where there is no need for photorealism, also because as we will see, the ***glossy*** effect is demanding as far as rendering time is concerned. In order to have a realistic effect one must use ***glossy*** without any tricks, thus ensuring an excellent final result.



■ **Figure 5.13**
Simulation of blurred reflections with ***VRayMtl***.

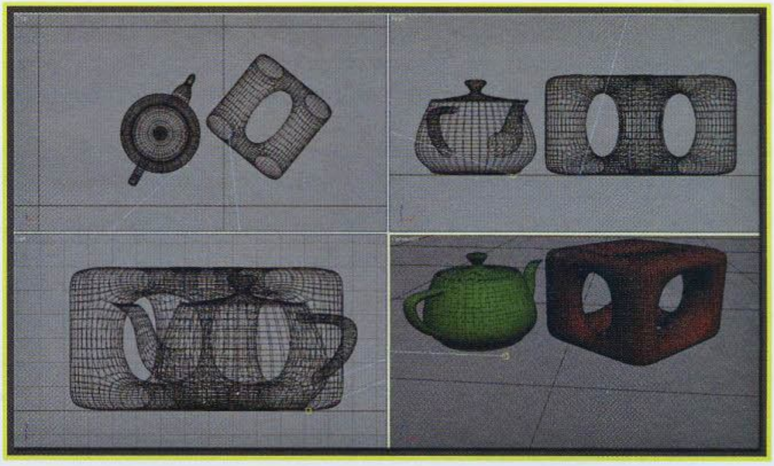
These are the 3 stages of reflection which a material can take on. There is a spread reflection when materials have an irregular enough surface for reflections to be absent. They can also be ***glossy***; this happens when reflections, although blurred, are visible. Finally, an object can be completely specular, like a mirror or chrome sphere.



■ **Figure 5.14**
Three real examples of reflections.

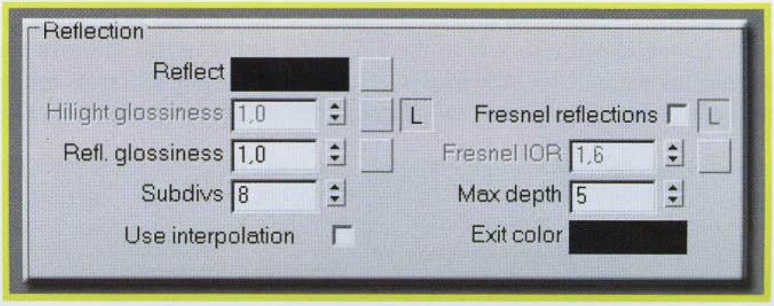
With ***VRay*** it is possible to obtain all these types, allowing the user to therefore choose every little detail.

For the analysis of parameters a very simple scene has been used, made of two geometrical objects placed on a plane and illuminated with the ***Physical Sky***.



■ **Figure 5.15**
Scene used for the analysis of the ***Reflection*** section.

The ***Reflection*** section has many more parameters than the ***Diffuse*** section, the first of these being the colored rectangle, which allows one to decide the amount of reflection to apply to the object.



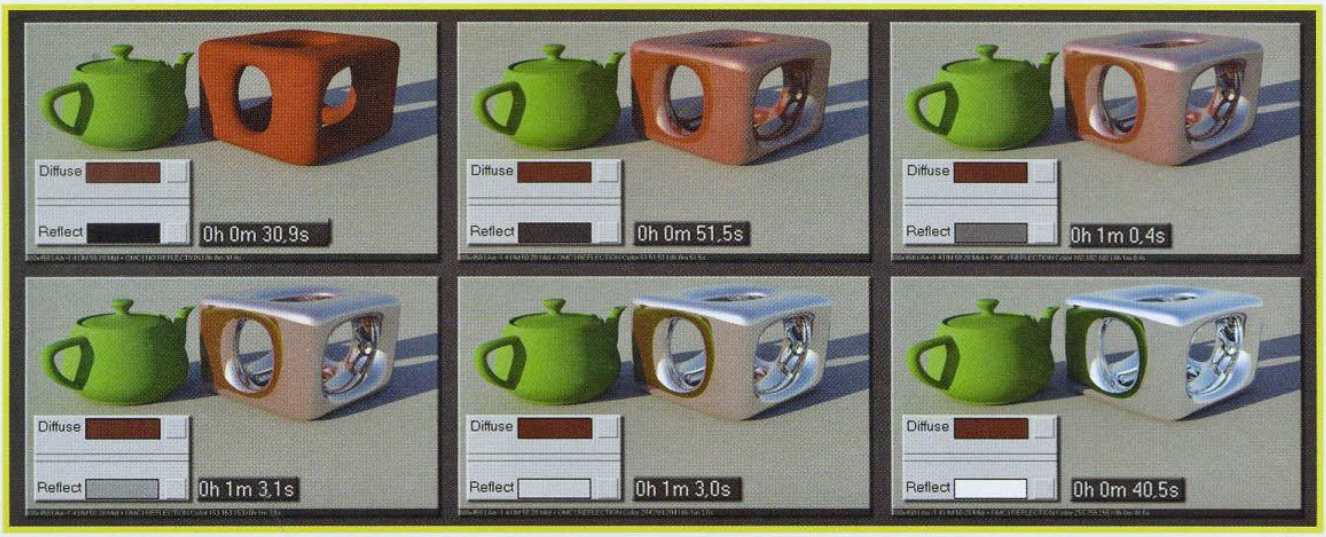
■ **Figure 5.16**
The ***Reflection*** section of the ***VRayMtl*** material.

Reflect - this box allows one to decide the amount of reflection for the material. The brighter the color is, the more the reflection increases. For chrome-effects, usually one keeps the ***Diffuse*** color very dark and changes the ***Reflect*** color from black to white. One can in any case increase or diminish reflections, not only by changing color from black to white but also by using colors. This way reflections can be colored.

■ **Figure 5.17**
Change in the color of reflection.

It is interesting to notice how by changing the ***Reflection*** channel from black to white increases rendering time, except with the color RGB = [255, 255, 255]. In this case rendering time undergoes a brusque decrease. Already at RGB=[254,254,254] the time is equivalent to 43 seconds.

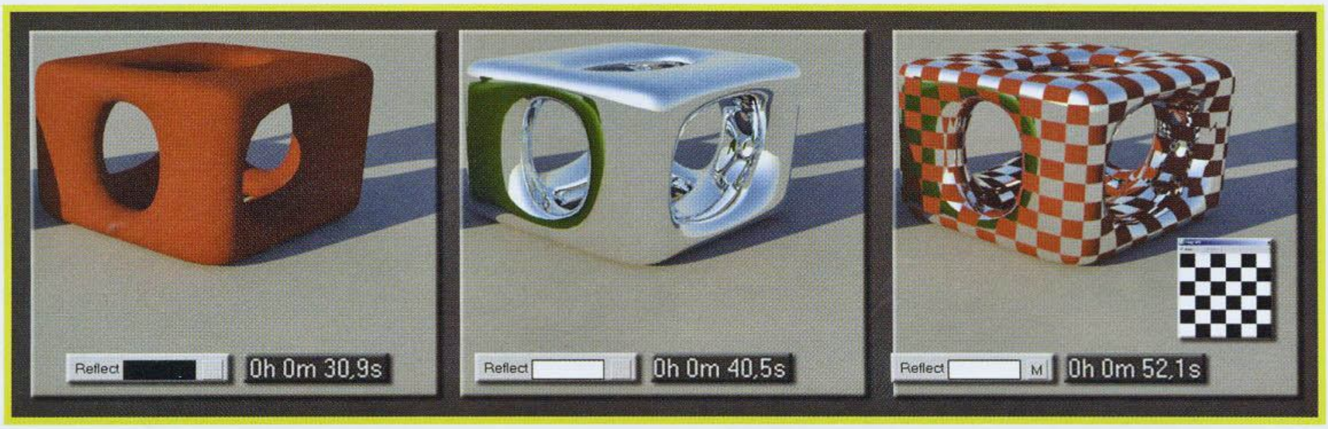
In the following example the reflection of the cube has been changed. This allows one to notice the behavior of ***Diffuse*** color after the variation of reflection.



■ **Figure 5.18**
Variation of reflection color on a colored object. In order to obtain colored reflections it is enough to change the color of the ***Reflect channel***.

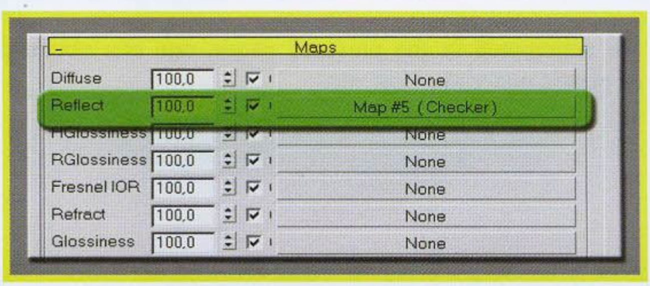
In this case the geometrical object is much more complex than the plane used previously. Rendering time has more than doubled with the use of reflection. Also in this case, with a completely white reflection, rendering time decreases compared to values between 1 and 244. One can observe another peculiarity: the variation of the object's color. Whereas initially it is brick red, as reflection is increased the color tends to disappear to give way to pure reflection. This phenomenon is called "Energy preservation", a concept which will be discussed later on.

Next to the color slots there is a square button which allows one to insert a map for defining areas which are supposed to reflect and others which are meant to be opaque.



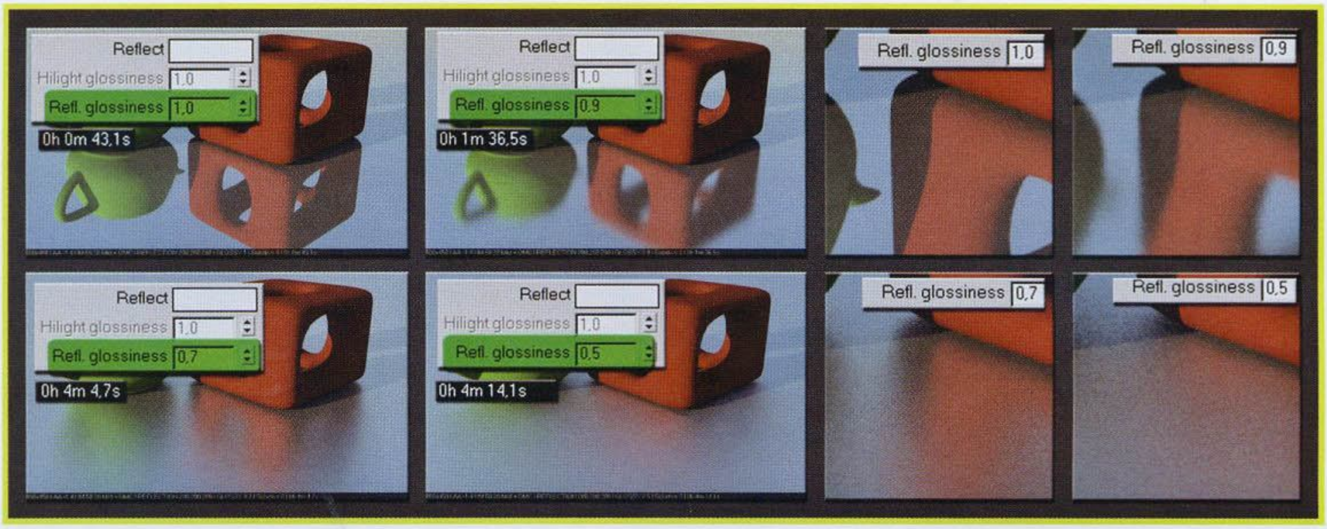
■ **Figure 5.19**
In order to determine which the superficial areas to be made reflective are, a black and white map has been applied in the reflection channel.

In the ***Maps*** rollout, the Checker map has now appeared in the ***Reflect*** channel. With the nearby spinner one can define the application intensity of that map. With the spinner set to 0, the map is not considered and the object is without reflections, providing the color of the ***Reflect*** channel is black.



■ **Figure 5.20**
Automatic insertion of the ***Maps*** rollout of the Checker map in the ***Reflect*** channel.

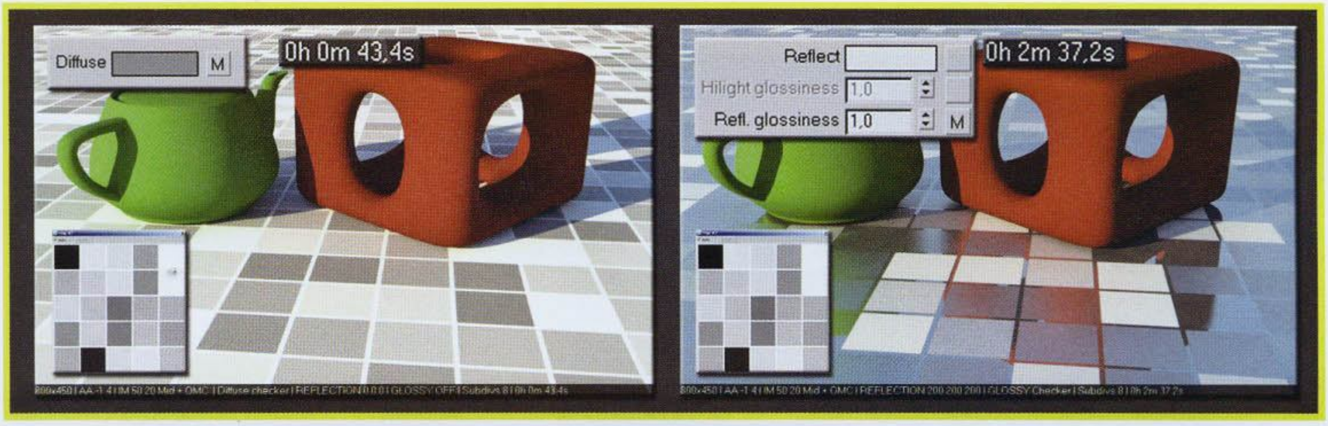
Refl. glossiness - this controls the amount of blurriness in reflection. Value 1.0 indicates a perfect reflection, typical of chrome materials. Below value 1.0 starts to blur reflections.



■ **Figure 5.21**
Change in the ***Reflection glossiness*** value.

Like in the photo of the mall we saw earlier, also with *VRay* one can create blurred reflections. Unfortunately rendering time increases greatly as blurring increases. In the reflection there is also a certain "grainy effect" which can be eliminated by using more ***Subdivisions***.

In this case there is a square button next to the ***Refl. Glossiness*** which allows one to insert a map which allows one to specify whether the amount of ***glossy*** effect to apply to reflective areas.



■ **Figure 5.22**
Definition of the ***Refl. Glossy*** channel by means of a 2D procedural map.

By using a black and white map, the white areas are perfectly reflective. Grey colors, all the way to black, allow one to define the intensity of the blends.

Hilight. glossiness - this controls the intensity of ***Highlights*** made without the use of raytrace reflections.

With the ***glossy*** effect, as we anticipated in the previous pages, one can obtain physically accurate ***Highlights*** of different intensities, typical of plastic materials, skin, etc...

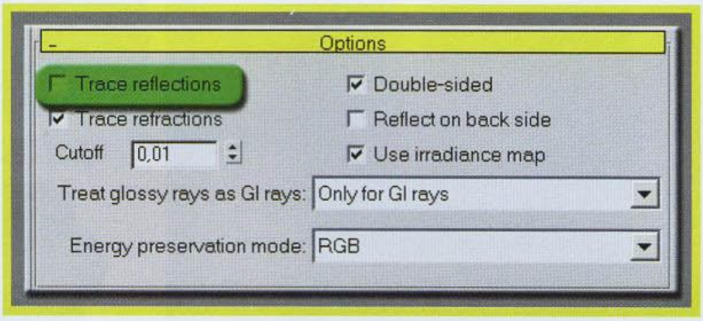
■ **Figure 5.23**
Highlight effect in real life.

By using an adequate calibration of the ***Reflect*** and ***Refl. Glossiness*** parameters one can obtain the same effects found in nature also within ***VRay***. These are none other than materials with low reflection and high ***glossy*** effect.

■ **Figure 5.24**
Creation of ***Highlights*** through use of the ***glossy*** effect.

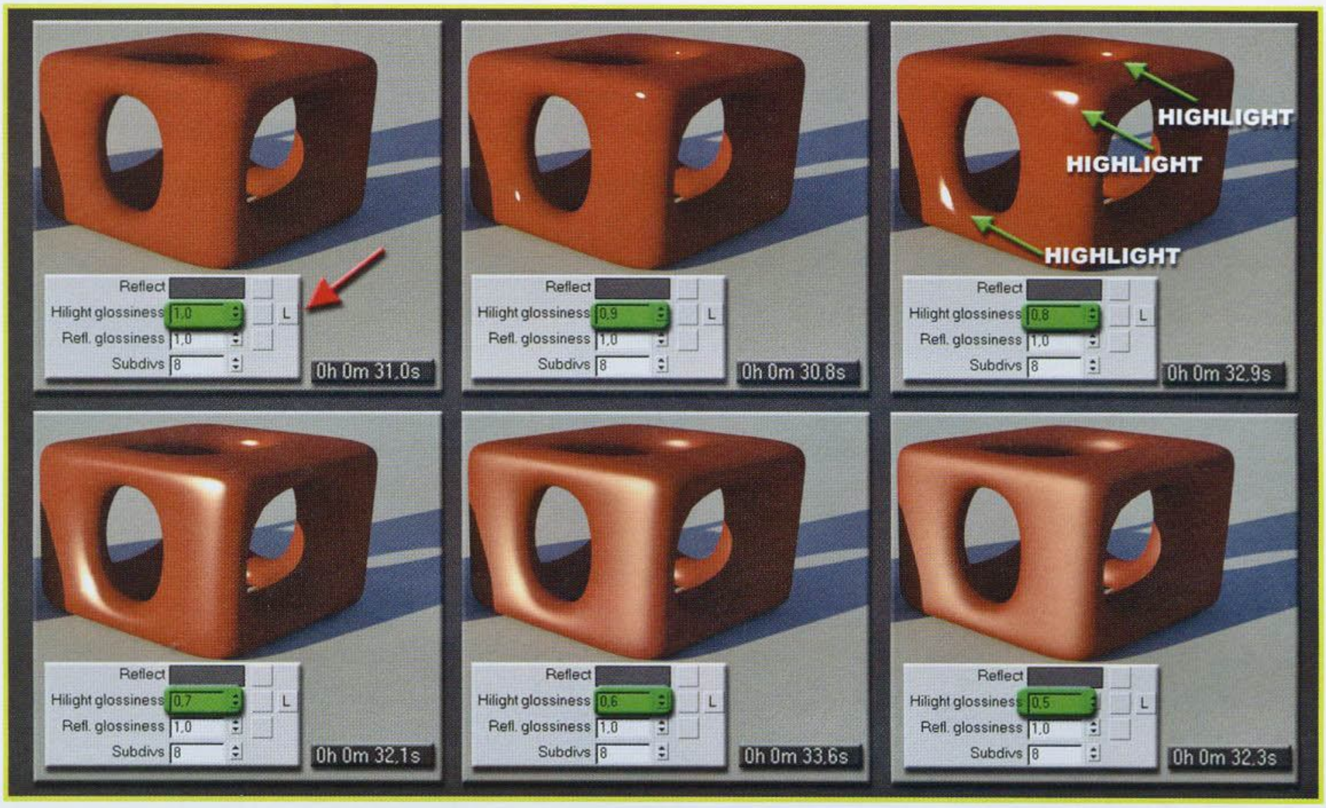
The generation of ***Highlights*** on curved geometries is generated by the reflection of the light source, in this case the Sun, on the 3D model. Increasing the ***glossy*** effect means that its reflections are spread on a wider surface, making ***Highlights*** of a greater extension. In this case ***Highlights*** are physically accurate because they are generated thanks to the reflection of an object on a surface with blurred reflections. Also in this example rendering time has doubled.

With *VRay* one can create "fake" ***Highlights*** as we have seen with the Standard materials of 3ds Max. In this case the color of reflections should be different to black in any case and in the ***Options*** rollout one must disable the ***Trace reflections*** option. This way reflections are not calculated on the material even though the color in the ***Reflection*** channel allows it.



■ **Figure 5.25**
Deactivation of the ***Trace reflections*** option.

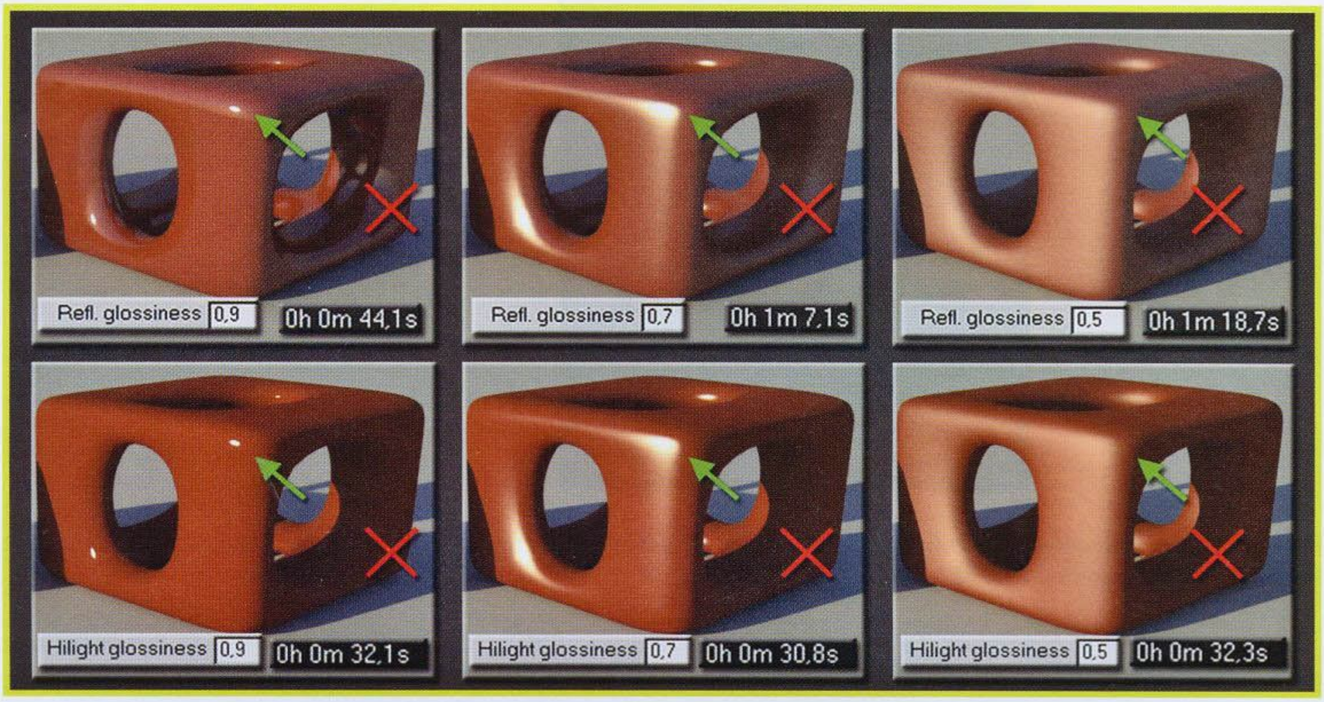
At this point one must activate the ***Highlight glossiness*** option by clicking on the letter **L** (***Lock***). The lower the value is, the more "fake blurring" there will be.



■ **Figure 5.26**
Variation in the ***Glossiness*** parameter.

Although ***Highlights*** are present, reflections are completely absent and some details which were to be found with real ***glossy*** reflections are no longer there. Also rendering time is the same because reflections are not being calculated.

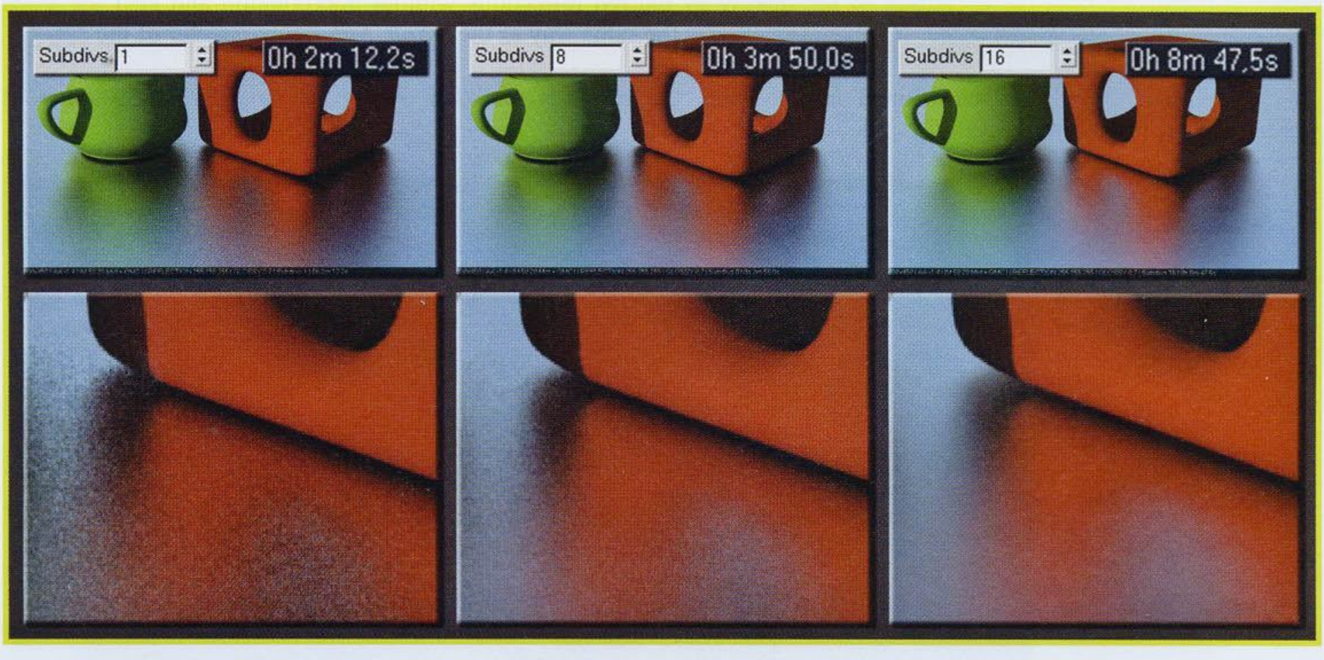
In these two examples, the one with physically correct ***Highlights*** and the one with fake ***Highlights***, the same reflection color has been used as well as identical ***glossiness*** values. In a direct comparison one can observe an interesting phenomenon:



■ **Figure 5.27**
Difference between ***Refl. Glossiness*** and ***Highlight glossiness*** parameters.

In a direct comparison one can observe how ***Highlights*** are identical in both cases. What changes is whether or not there are blurred reflections, which increase rendering time. It is obvious that the first case is the one that ensures higher quality at the cost of more time. If for example the aim is that of creating plastic materials one might consider using the second method with "false" ***Highlights***. Also in this case there is a square button which allows one to specify, via a map, the exact position of ***Highlights***.

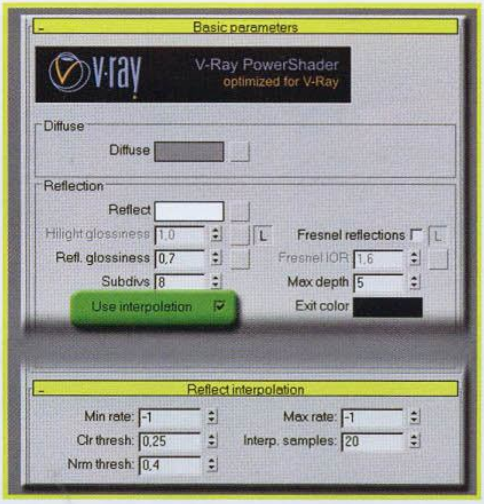
Subdivs - this controls the quality of ***glossy*** reflections. High values produce better images and require more time. Low values make renders faster by producing blurred reflections of low quality with a lot of noise.



■ **Figure 5.28**
Variation of the ***Subdivision*** for reflections. As subdivisions increase, so does rendering time, noise decreases significantly and rendering quality is higher.
- - - DVD - - -

Use interpolation - in order to quicken ***glossy*** calculation, ***VRay*** provides the possibility to interpolate the calculation, exactly like the GI and the ***IM***. By activating this parameter, ***VRay*** automatically uses the information in the ***Reflect interpolation*** rollout for the calculation of reflections.

■ **Figure 5.29**
Activation of interpolation of reflections and the relative rollout.



For interpolated reflections there are parameters identical to those used for the ***IM***. For a more detailed explanation one should read the chapter on the ***IM***.

Min rate - this value determines the resolution of the first phase of ***glossy*** calculation. When set to 0, this means that the resolution of the first phase, which is the one at lowest resolution, will be the same as the final image resolution.

Max rate - this value determines the resolution of the last phase of ***glossy*** calculation.

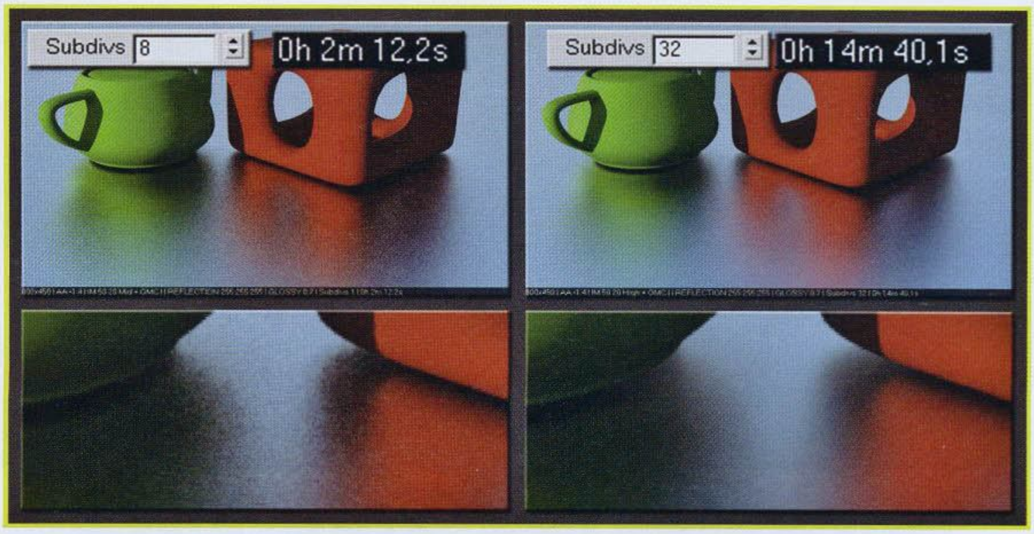
Clr thres - this parameter controls the sensitivity of the algorithm for ***glossy*** calculation towards variations of indirect light.

Nrm thres - this parameter controls the sensitivity of the algorithm for ***glossy*** calculation towards variations of the normals of surfaces or superficial details.

Interp sample - this parameter controls the quality of samples used for interpolation of samples used for ***glossy*** calculation and generated according to the ***Subdivs*** parameter in the ***Reflection*** section. The ***Subdivs*** parameter we saw earlier is useful for both the definition of the quality of blurred reflections without interpolation and also for when interpolated reflections are activated. It behaves in exactly the same way as the ***HSph. Subdivs*** parameter of the ***IM***.

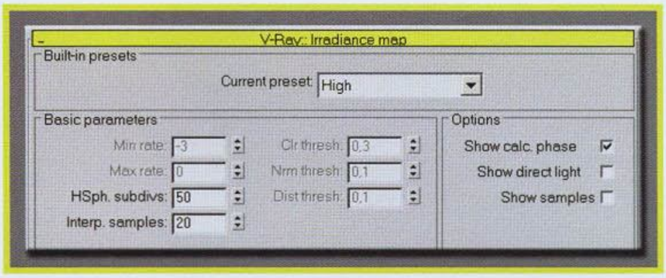
As the best quality is obtained without interpolation, the scene is rendered first of all with a high number of ***Subdivisions*** without interpolation.

■ **Figure 5.30**
Two renders with ***Glossy reflection*** set with default parameters (on the left). On the right, it has been setup with high ***Subdivisions***.



The renders of the previous image have been obtained without interpolation. The one on the right, because of its high quality, is used as a reference image. Through interpolation we will try to reach the same level of quality in less time. Meanwhile, with a series of examples, the function of the ***Reflect interpolation*** rollout parameters will be explained.

First of all, observe the setting of the ***IM***.



■ **Figure 5.31**
Parameters of the ***IM*** used for this test.

The ***High preset*** has been applied. 4 ***Prepasses*** will be applied, as explained in the chapter concerning the ***IM***. This is useful to know because by using interpolation for reflections, according to their ***Min rate*** and ***Max rate*** settings, it can happen that for them ***VRay*** calculates extra ***prepasses***. If for example the parameters of ***Reflect interpolation*** are ***Min rate*** = -1 and ***Max rate*** = 1, ***VRay*** carries out 5 ***Prepasses***, an extra one being due to ***Reflect interpolation*** settings. In the first two passes only the GI is calculated, whereas in the 3rd and 4th phase also ***glossy*** calculation is added. In the last phase only ***glossy*** is calculated.

Irradiance Map - ***Min rate***= -3 ***Max rate***= 0
Reflect Interpolation - ***Min rate***= -1 ***Max rate***= 1

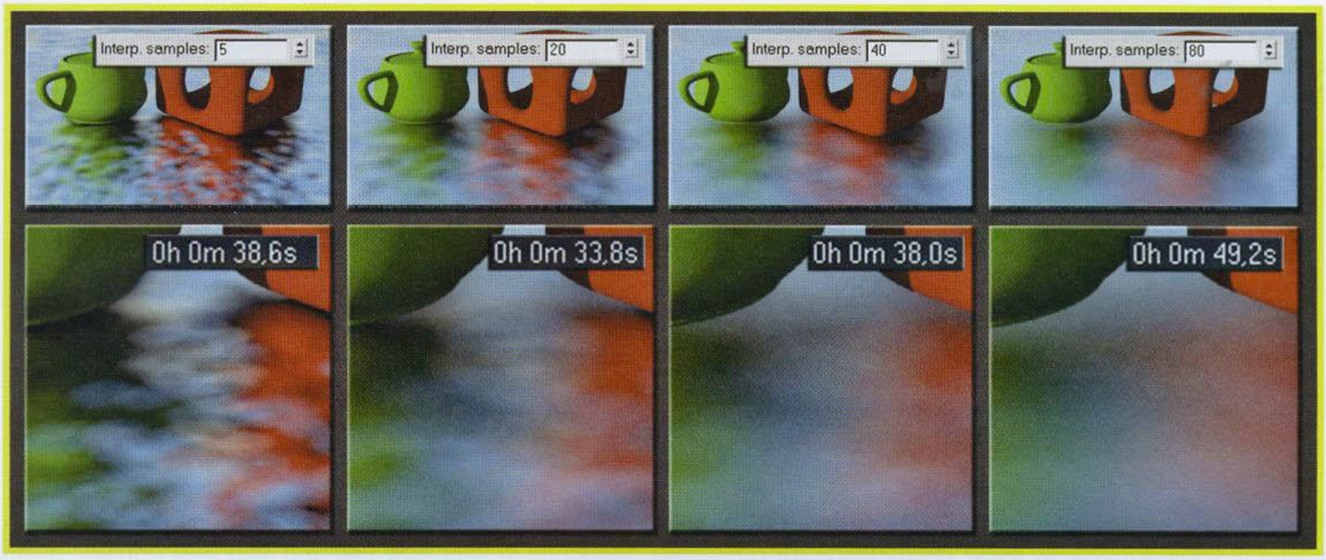
PREPASS	IRRADIANCE MAP	REFLECT INTERPOLATION
-3	X	
-2	X	
-1	X	X
0	X	X
1		X

Take this scheme only as an example, because usually one tends to use the same ***Min rate*** and ***Max rate*** values for both.

We will begin by using low values of ***Reflect interpolation*** so that the effects of changes in various parameters can be seen.

■ **Figure 5.32**
Use of low parameters in ***Reflect interpolation***.

The quality of reflections is obviously very low and one can notice samples used for ***glossy*** calculations. Also the ***Interp. samples*** parameter will be changed with low values.



■ **Figure 5.33**
Variation of the ***Interp. samples*** parameter in the ***Reflect interpolation*** rollout.

By increasing this parameter, for ***glossy*** calculations more and more samples are considered per surface unit. This would seem to improve quality by eliminating stains of single samples. At the same time one loses detail and both the teapot and the box appear to be floating, just like what would happen to the ***IM*** if ***Interp. samples*** were too high. The default value of 20 is a good compromise between quality and loss of detail.

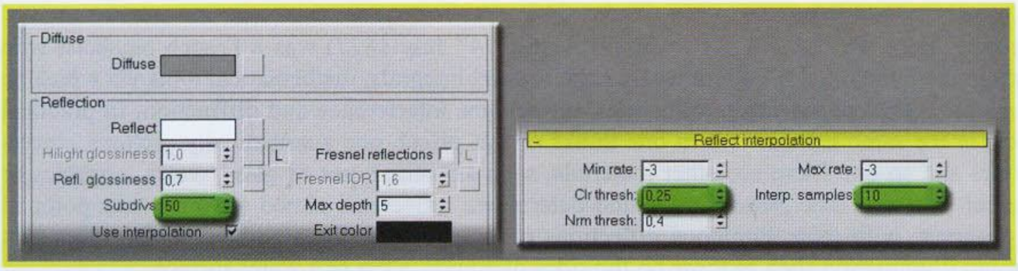
Now also the ***Subdivs*** parameter will be modified from a low value, with ***Interp. samples*** still set to 5.

■ **Figure 5.34**
Variation of the ***Subdivs*** parameter applied to interpolation.
- - - DVD - - -

The increase of ***Subdivision*** values, already at low levels, brings a distinct improvement in quality. In this case, a value between 20 and 50 is the best solution. Rendering time, obviously, has worsened, with an increase of 25%.

Default values for ***Clr thresh*** and ***Nrm thresh***, which behave in exactly the same way as they do for the ***IM***, ensure satisfactory results for most situations, unless one is using interpolation on particularly complex geometries.

The last two parameters to be analyzed are ***Min rate*** and ***Max rate***. For other parameters medium-high values will be used.



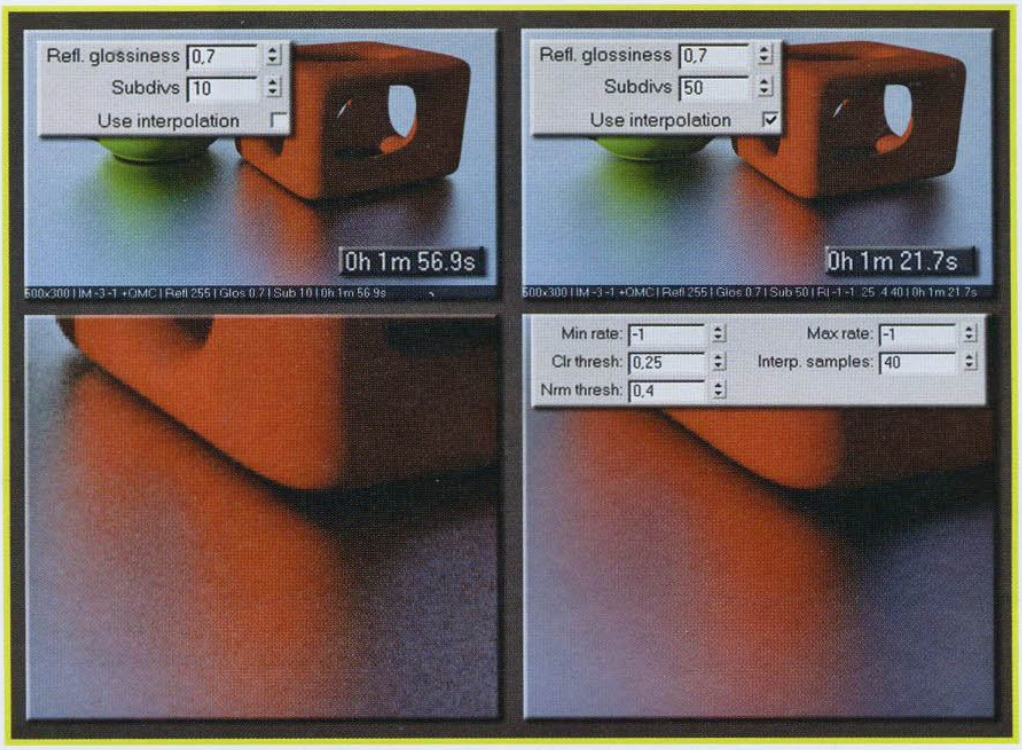
■ **Figure 5.35**
Use of medium-high parameters of ***Reflect interpolation***.

By default they are set to ***Min rate***= -1 and ***Max rate***= -1.

■ **Figure 5.36**
Identical variation of the ***Min rate*** and ***Max rate*** parameters. Time demand has increased but also quality and precision of ***glossy*** reflections.
- - - DVD - - -

In fact this is the last setting which ensures low rendering times and good quality

To sum up, one can say that interpolation of ***glossy*** effects is an excellent method for saving time. One of its main flaws is the chance of it producing flickering during animations, although this problem can be solved with higher values. By using the direct method flickering is always absent, even with low values.



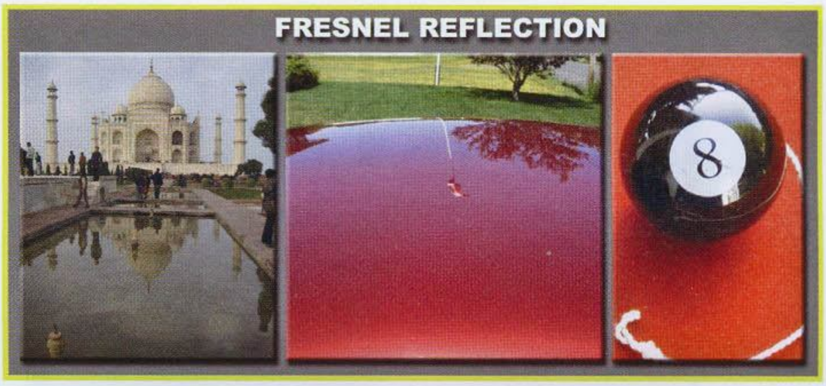
■ **Figure 5.37**
Comparison between the direct method and the indirect one for blurred reflections.

A fundamental rule is the coherence between the ***Min rate*** and ***Max rate*** parameters of the ***IM*** and the ***RI***. For example, if ***Min rate***= -1 and ***Max rate***= 0 for ***RI***, then also the ***IM*** must have ***Max rate***= 0, or the GI will not be correctly represented in blurred reflections. In other words the GI will not be reflected.

Also the ***RI*** cannot be generated correctly if the ***IM*** has been loaded and therefore not calculated on-the-fly, but generated for example with the option ***Incremental add to current map***.

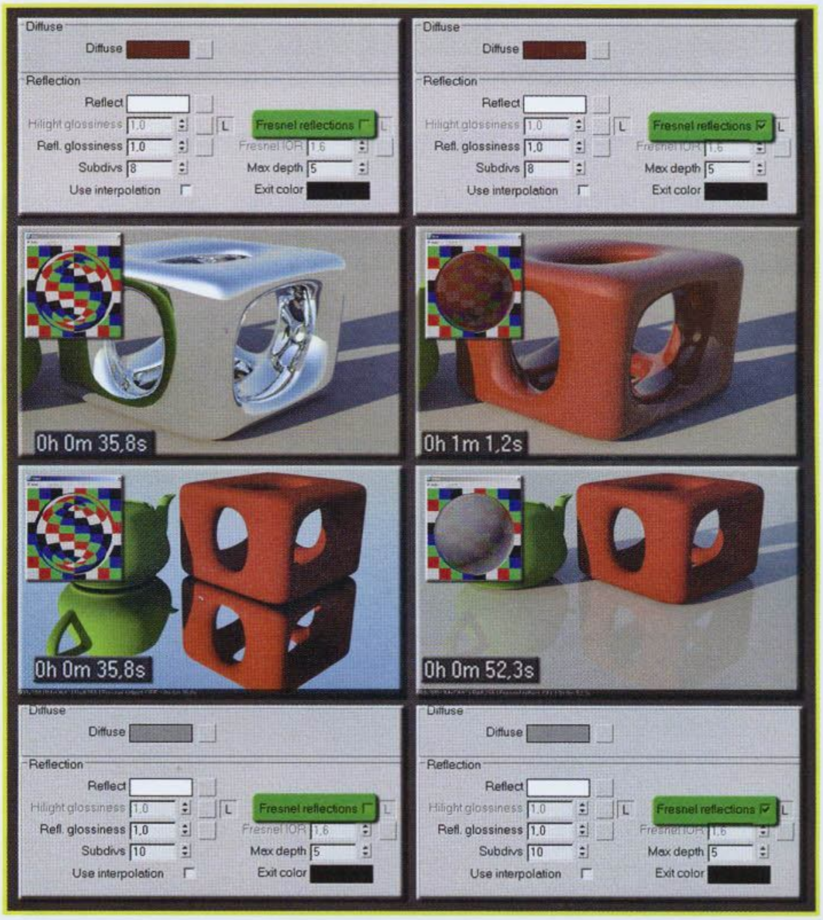
Fresnel reflection - by activating this option, reflection depends on the angle one observes the object from. Only some materials have this particular feature; ***fresnel*** reflection also depends on the **IOR** refraction index of the material.

Augustin Jean Fresnel (Broglie, Eure, 10th May 1788 – Ville d'Avray, 14th July 1827) was a French physicist who carried out important research in the optical field. To him we owe the Fresnel integrals, mathematical tools via which he found a theory which explains all optical phenomena: reflections, refractions, interference and diffraction. In practical terms, not all material reflects in the same manner. When one thinks of this effect, one automatically pictures objects like chrome surfaces or mirrors and so on. There are other materials, such as some types of glass, water, ceramics or plastic which follow a different behavior. The ***fresnel*** reflection determines reflections aimed towards ones point of view, perpendicular to the observer, with more luminous reflections on angle faces, faces parallel to the viewpoint which create points of maximum illumination such as those on the sides of a glass, a car or a water surface.



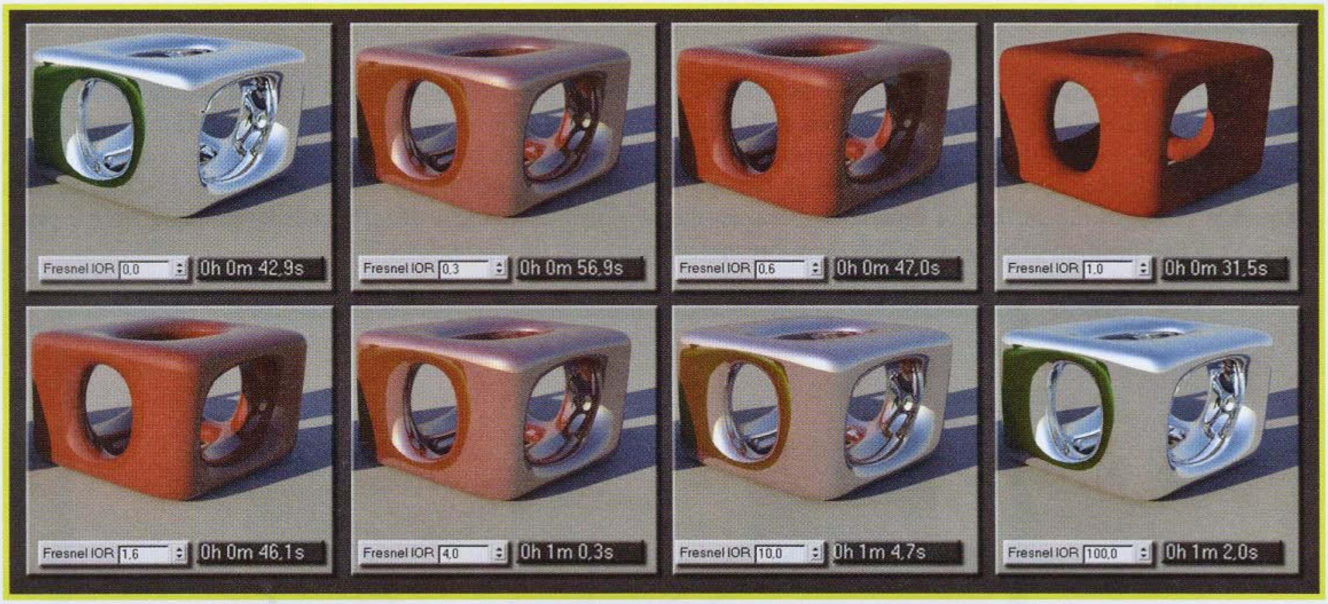
■ **Figure 5.38**
Fresnel reflections in nature.

VRay allows one to simulate this effect by activating the appropriate ***Fresnel reflection*** parameter.



■ **Figure 5.39**
Activation of the ***Fresnel reflections*** parameter. This parameter has been activated in a material with a completely white reflection color. One can see how even with such a high reflection, the ***Fresnel*** attenuates the effect by accentuating it in curved geometries, in a parallel way compared to the observer's point of view. In perpendicular faces, such as the front vertical surface of the cube, reflection is very low. As far as the plane is concerned instead, the ***Fresnel*** effect is not as obvious as it is on the cube. Rendering time increases considerably. Also the preview of the 3ds Max Material Editor gives us a correct vision. This allows us to make a first estimate of what the material will be like in the final render.

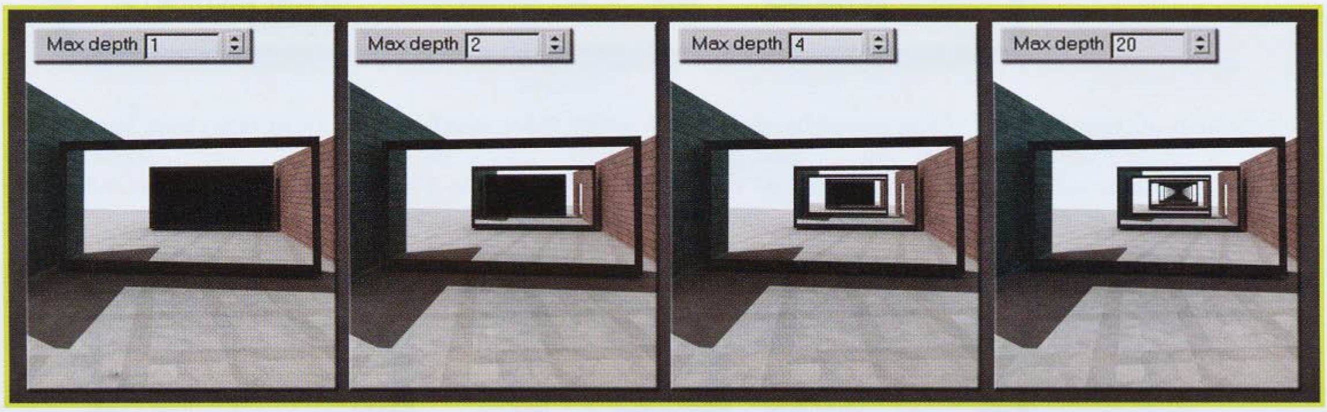
By default, the IOR of the ***Fresnel*** is blocked. By clicking the **L** (***Lock***) button one can specify its value manually.



■ **Figure 5.40**
Variation of the ***Fresnel IOR*** parameter. From 0 to 1 the effect goes from a perfect chrome effect to a complete absence of reflection until it slowly increases up to a maximum value of 100.

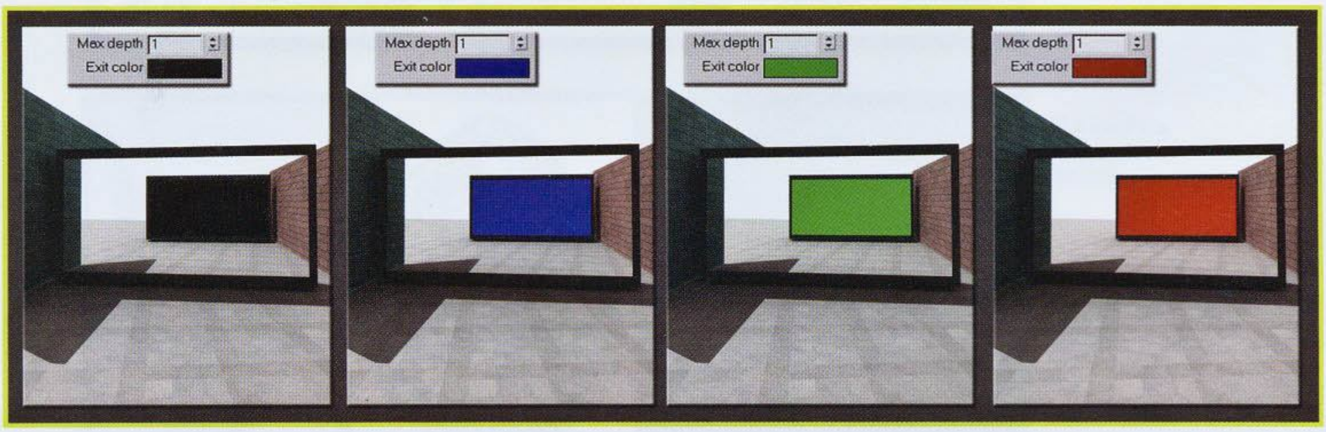
As before, also in this case one can use a map to be used in the ***Fresnel IOR*** channel.

Max depth - this represents the number of times that rays can be reflected. For a correct render, scenes with lots of objects with reflection properties require high values.



■ **Figure 5.41**
By changing the ***Max depth*** parameter one can specify how many times an object can be reflected. This parameter is found also in the ***VRay: Global switches*** rollout.

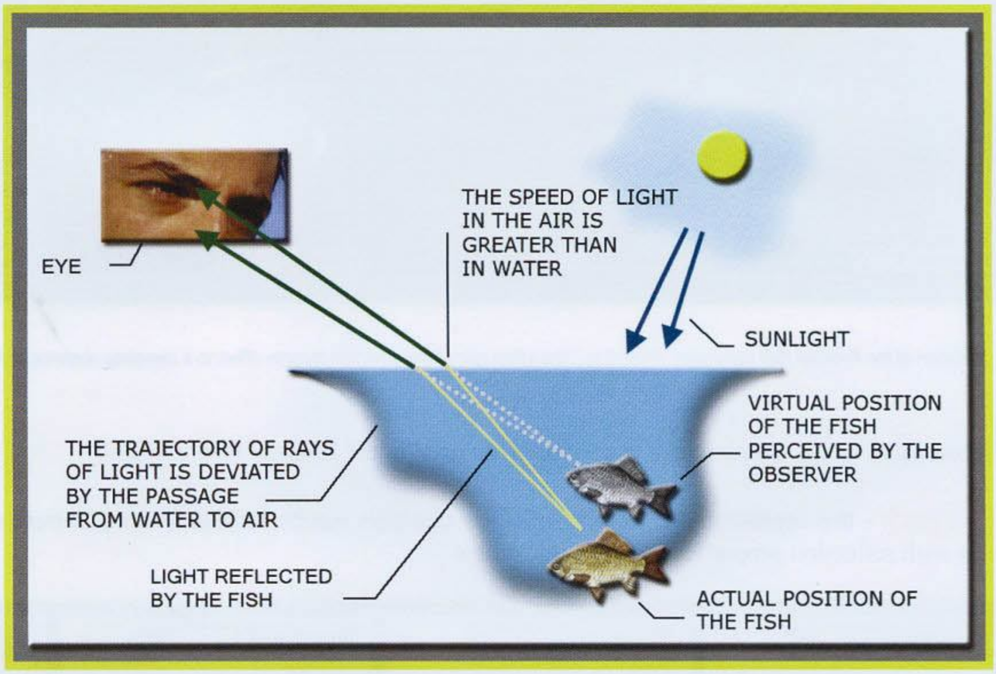
Exit color - when rays of the reflection reach the number of bounces set via the ***Max depth*** parameter, the geometry uses this color to represent reflections.



■ **Figure 5.42**
By changing object ***Exit color***, when the number of reflections ends, it takes on the color shown in the box.

REFRACTION

When a ray of light crosses through different materials, it undergoes the phenomenon of refraction, which consists of a variation of speed and direction of propagation of light. The deviation of the beam, in other words the amplitude of the angle of refraction, depends on the ratio between the speed of light within the two different mediums. For example, a beam of light reflected by a fish in the sea, when passing from the water into the air, increases its speed of propagation and moves closer to the surface of separation between the two mediums. Light therefore seems to come from a point above the actual position of the fish, which therefore appears closer to the surface.



■ **Figure 5.43**
Refraction scheme.

Each medium of propagation is characterized by an **IOR** refraction index which, except for in rare cases, increases as the density of the medium increases. Air for example has an IOR which is not much greater than 1, whereas 1 by definition represents vacuum. Water has an IOR of 1.33, glass is equivalent to 1.4 for the least dispersive, and 1.7 for the denser types. In diamonds, the most dispersive known in nature, the IOR reaches 2.4.

■ **Figure 5.44**
Some *IOR* values.

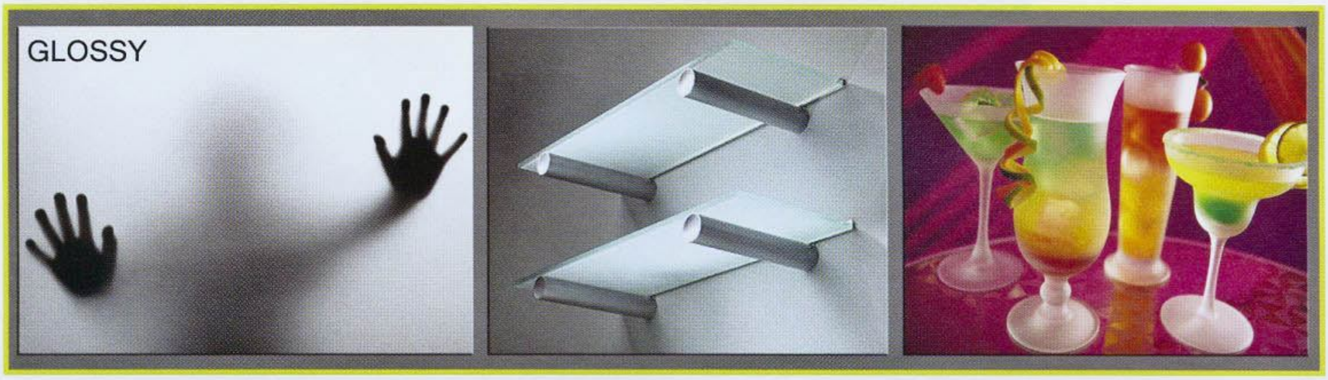
Material	λ (nm)	*IOR*
Vacuum		1 (exactly)
Air @ STP		1.0002926
Gases @ 0 °C and 1 atm		
Air	589.29	1.000293
Helium	589.29	1.000036
Hydrogen	589.29	1.000132
Carbon dioxide	589.29	1.00045
Liquids @ 20 °C		
Benzene	589.29	1.501
Water	589.29	1.333
Ethyl alcohol (ethanol)	589.29	1.361
Carbon tetrachloride	589.29	1.461
Carbon disulfide	589.29	1.628
Solids at room temperature		
Diamond	589.29	2.419
Strontium titanate	589.29	2.41
Amber	589.29	1.55
Fused silica	589.29	1.458
Sodium chloride	589.29	1.50
Other materials		
Pyrex (a borosilicate glass)		1.470
Ruby		1.760
Water ice		1.31
Cryolite		1.338
Acetone		1.36
Ethanol		1.36
Teflon		1.35 - 1.38
Glycerol		1.4729
Acrylic glass		1.490 - 1.492
Rock salt		1.516
Crown glass (pure)		1.50 - 1.54
Salt (NaCl)		1.544
Polycarbonate		1.584 - 1.586
PMMA		1.4893 - 1.4899
PETg		1.57
PET		1.5750
Flint glass (pure)		1.60 - 1.62
Crown glass (impure)		1.485 - 1.755
Fused Quartz		1.46
Bromine		1.661
Flint glass (impure)		1.523 - 1.925
Cubic zirconia		2.15 - 2.18
Diamond		2.419

■ **Figure 5.45**
Real examples of refraction for completely transparent materials.

As well as these classic examples of refraction, there are many others, like the refraction of colored objects with variable sections. In this case refraction causes objects to have a more intense color in areas which are thicker compared to the thinner ones.

■ **Figure 5.46**
The variation of color intensity in glass is due to the thickness and size of the object.

Another phenomenon due to refraction is that of glazing. A glass or a similar object with a high level of superficial irregularity can generate blurred refractions.



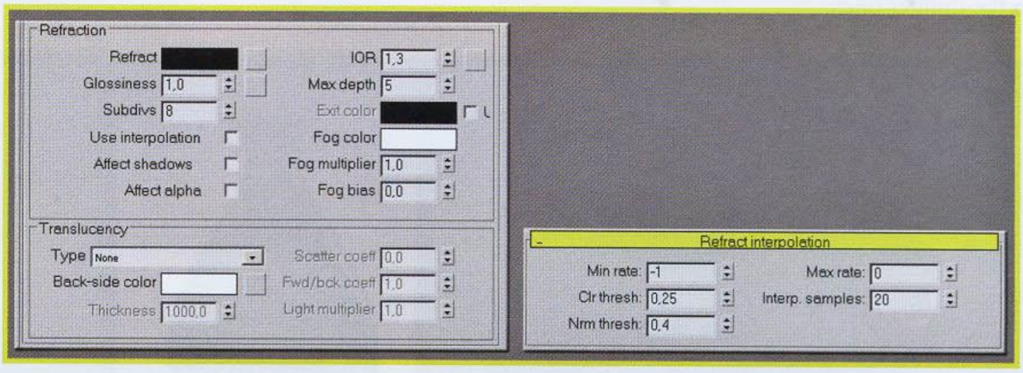
■ **Figure 5.47**
Examples of glazed glass, also called frosted glass.

Another not so well-known phenomenon is so-called ***Subsurface scattering*** or in jargon, ***SSS***.

■ **Figure 5.48**
Examples of ***Subsurface scattering***.

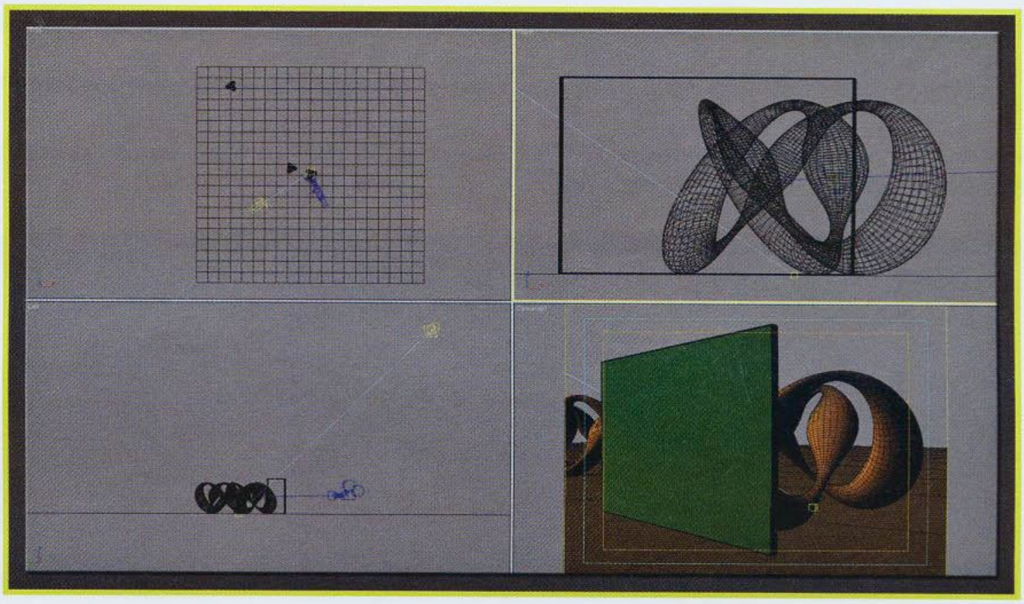
SSS is the process which allows light to spread through the material of an object. Human skin, wax, marble and milk are some of the materials which possess this characteristic.

All these physical behaviors of light are perfectly and realistically reproducible within *VRay*. The next chapter analyzes each parameter of the ***Refraction*** section, with some example tests carried out to help understand this process.

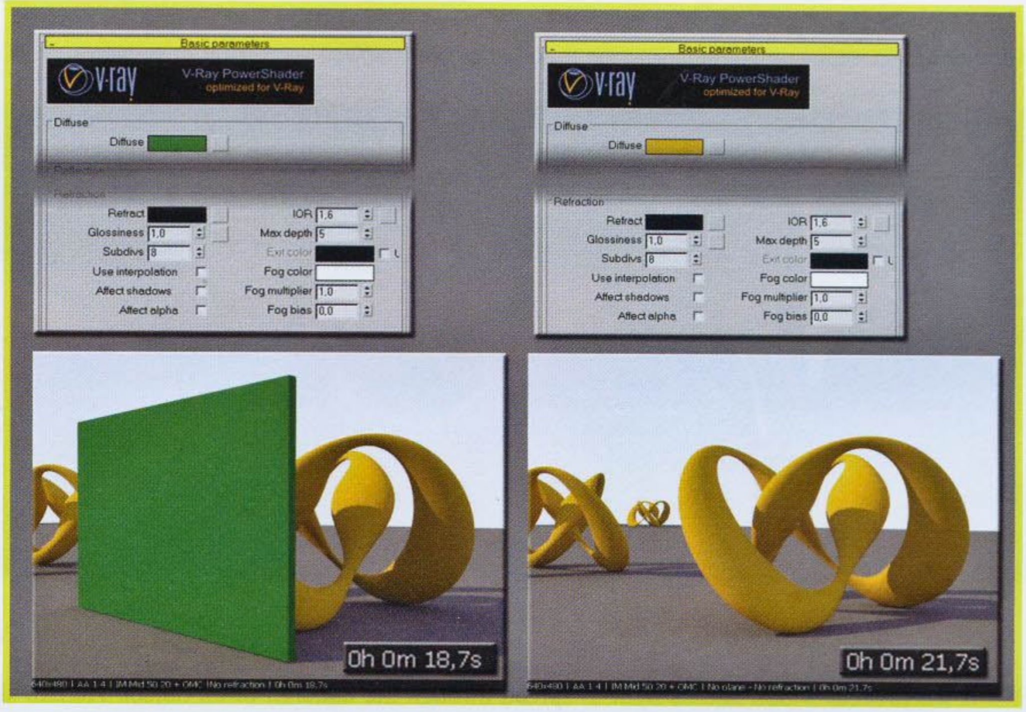


■ **Figure 5.49**
The sections dedicated to refraction: ***Refraction***, ***Translucency*** and ***Refract interpolation***. This last section should be familiar.

The scene used is made up of four geometries, three Torus Knots, a Chamfer box and a plane, all illuminated with a system made up of ***skylight*** + ***Direct Light***.



■ **Figure 5.50**
Scene used for the explanation. Notice how the rectangular plane has been placed in front of objects with varying thickness. This particular array has been used for the demonstration of both the frost effect and for ***Fog color***.



■ **Figure 5.51**
Render without refractions or reflections. The material only has a color. In the next test there will be no reflection. It will be used only to better evaluate the effects of refraction.

Refract - the color of this box controls the quantity of refraction of the material.

■ **Figure 5.52**
By changing the color of refractions from black to white, transparency increases.

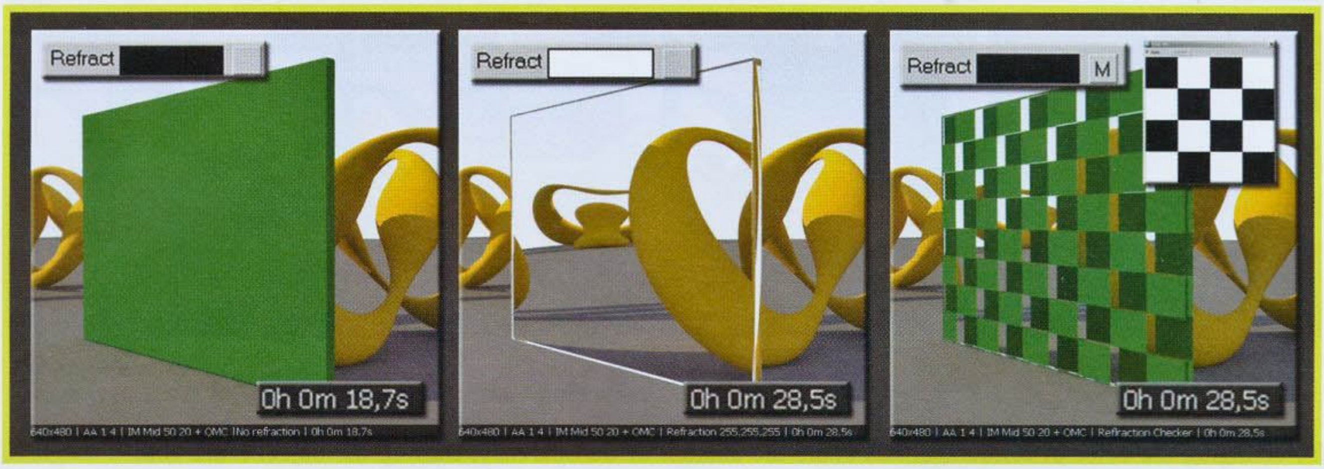
By changing the color of refractions one alters the amount of transparency, making the object more similar to glass. Gradually the green ***Diffuse*** color disappears. Similarly to reflection, also for refraction rendering time increases until the color threshold of RGB = [254, 254, 254]. With RGB = [255, 255, 255] rendering time drops drastically.

It is however possible to assign a color to refraction. This way not only does one change the intensity of refraction but also the object's color.

■ **Figure 5.53**
Different results with different refraction colors.

The color of refractions can generate other effects also according to the ***Diffuse*** color. It is advisable to set ***Diffuse*** color to black if one wishes to use a color for the ***Refract*** channel: the result will then be predictable but most of all correct.

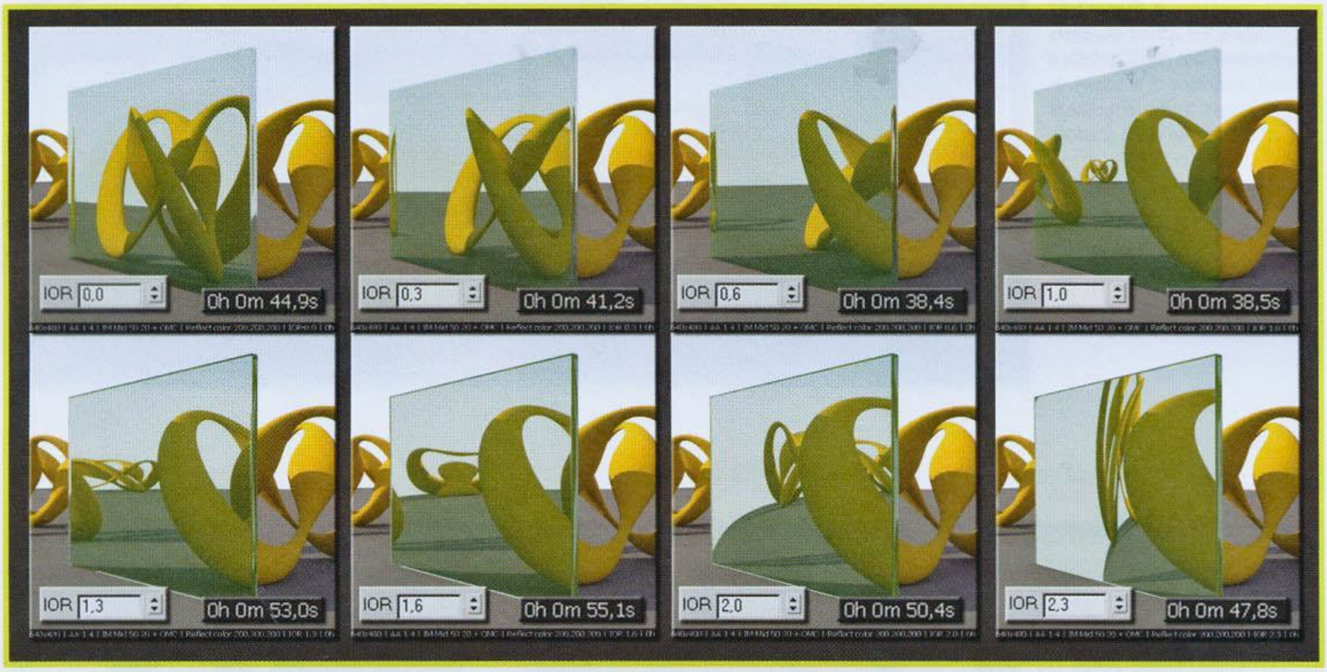
Just like reflection, also refraction can be defined by means of a black and white map, by clicking on the button just right of the color.



■ **Figure 5.54**
Definition of refraction via a procedural map.

IOR - this represents the refraction index of the material. The optical properties of a medium are quantified by a parameter called absolute index of refraction. It consists of a pure number, which characterizes each medium and is equivalent to 1 for vacuum and higher values for every other material medium. It represents the reduction factor of the speed of propagation of light and any other form of electromagnetic radiation, when it travels through any medium which is not empty space.
V = c/n, where there is the speed of light through vacuum (about 300,000 km/s) and it is the absolute index of refraction for the considered medium. Usually the absolute index of refraction for gas does not go much beyond 1 (air is at 1,000293), liquid and solid materials vary from a minimum of 1.3 to a maximum of about 2.8.

Steel	2.50	Beryl	1.577	Iron	1.51	Plexiglas	1.50
Acetone	1.36	Berilonite	1.553	Flourite	1.434	Polyethylen e	1.55
Water (gas)	1.0002	Bromine (liquid)	1.661	Formica	1.47	Quartz	1.544
Water 100' C	1.318	Bronze	1.18	Stone	1.660	Copper	1.10
Water 20' C	1.333	Calcite	1.486	Chalk	1.510	Ruby	1.760
Water 35' C	1.331	Cassiterite	1.997	Ice	1.309	Rutile	2.62
Aquamarine	1.577	Celestite	1.622	Jade	1.610	Sepiolite	1.530
Turpentine	1.472	Cerussite	1.804	Glycerine	1.473	Serpentine	1.560
Adularia	1.525	Cyanite	1.715	Rubber	1.519	Sylicon	4.24
Agate	1.544	Citrine	1.550	Graphite	2.01	Emerald	1.576
Musky agate	1.540	Chlorine (gas)	1.0007	Hydrogen(gas)	1.0001	Steathite	1.539
Alcohol	1.329	Chlorine (liq)	1.385	Hydogen (liq)	1.097	Stronzium	2.410
Ethyl alcohol	1.36	Chloride (salt)	1.544	Iolite	1.548	Tantalite	2.240
Alexandrite	1.745	Blue Cobalt	1.74	Leucite	1.509	Tanzanite	1.691
Alluminium	1.44	Green Cobalt	1.97	Magnesium	1.515	Teflon	1.35
Amber	1.546	Purple Cobalt	1.71	Malachite	1.655	Topaz	1.620
Amethyst	1.544	Shell	1.530	Mercury (liq)	1.62	Tremolite	1.600
Anastasium	2.490	Coral	1.486	Methanol	1.329	Turquoise	1.610
Andalusite	1.641	Corindone	1.766	Moldavite	1.500	Glass	1.517
Anhydrite	1.571	Cystal	2.00	Nylon	1.53	Sapphire	1.760
Apatite	1.632	Iodium crystal	3.34	Olivine	1.670	Sulphur	1.960
Aragonite	1.530	Yellow chrome	2.31	Onyx	1.486		
Silver	1.3	Red Chrome	2.42	Opal	1.450		
Argon	1.0002	Green chrome	2.4	Gold	0.47		
Air	1.0002	Diamond	2.417	Obsidian	1.489		
Asphaltite	1.635	Jasper	1.540	O2 (gas)	1.0002		
Ivory	1.540	Dolomite	1.503	O2 (liquid)	1.221		
Azurite	1.730	Ebony	1.66	Periclasium	1.740		
Barite	1.636	Helium	1.00003	Pearl	1.530		
Benitoite	1.757	Emathyst	2.940	Pyrite	1.810		
Benzene	1.501	Ethanol	1.36	Plastic	1.460		



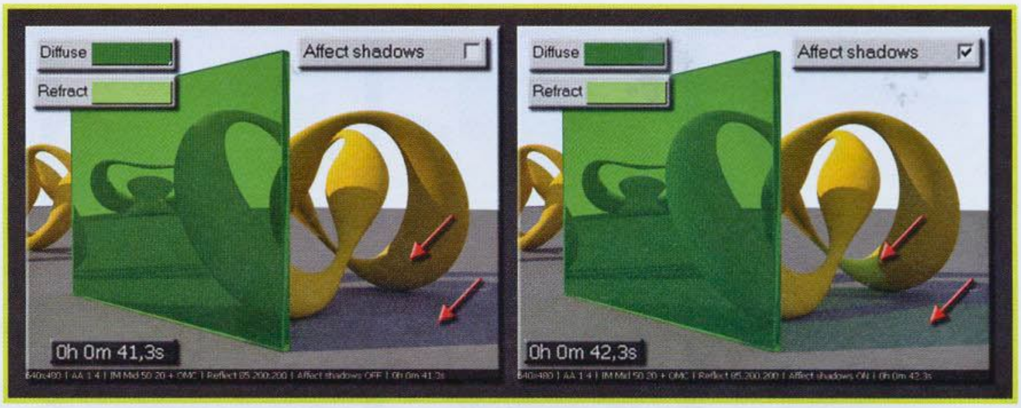
■ **Figure 5.55**
Glass wall with various ***IOR*** values.

By changing the ***IOR***, transparent objects deviate light because of the different density of materials and with high settings they create strange deformations which can be obtained only with materials like diamond or crystal. It should be noticed how with ***IOR*** = 1 there is no type of deformation. This is because *VRay* treats the material as if it were made of air, just like the environment around it. Also in this case, the square button next to the spinner allows one to input a map for ***IOR*** definition.

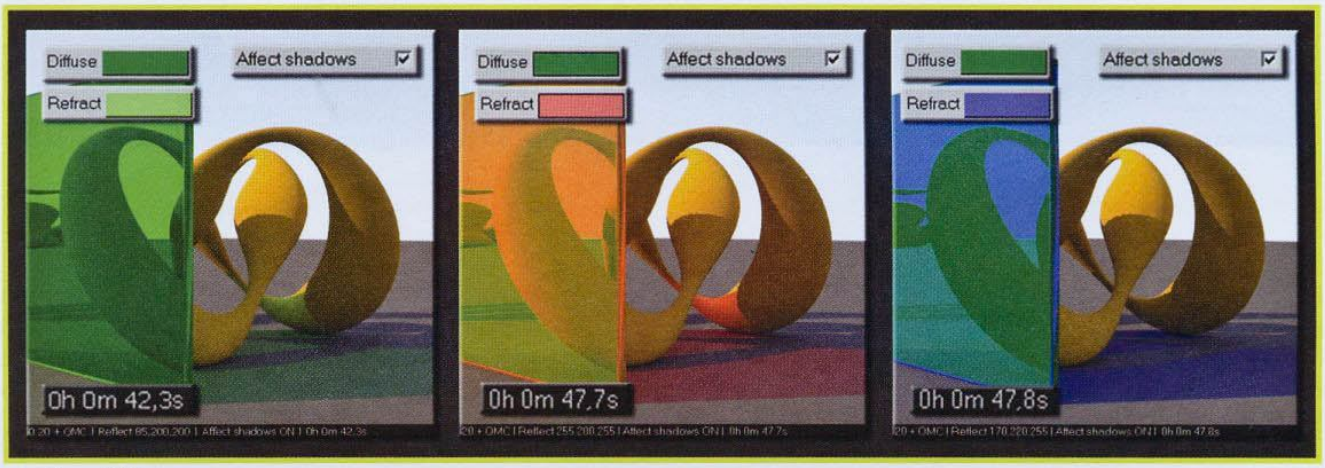
Affect shadows - activating this parameter for refraction causes the object to produce transparent shadows instead of solid ones. Until now the shadow produced by the vertical panel has been solid. If one observes a glass object and its shadow instead, one will see that the latter is not dark but transparent and colored. This effect works only with *VRay* shadows.

■ **Figure 5.56**
Shadows generated by transparent objects are never solid but can be colored if the material of the object is colored.

■ **Figure 5.57**
Thanks to the ***Affect shadows*** option the glass plane permits a correct generation of shadows. In this case, as the refraction is colored, so is the shadow.



Notice how the color of refraction influences the color of shadows.

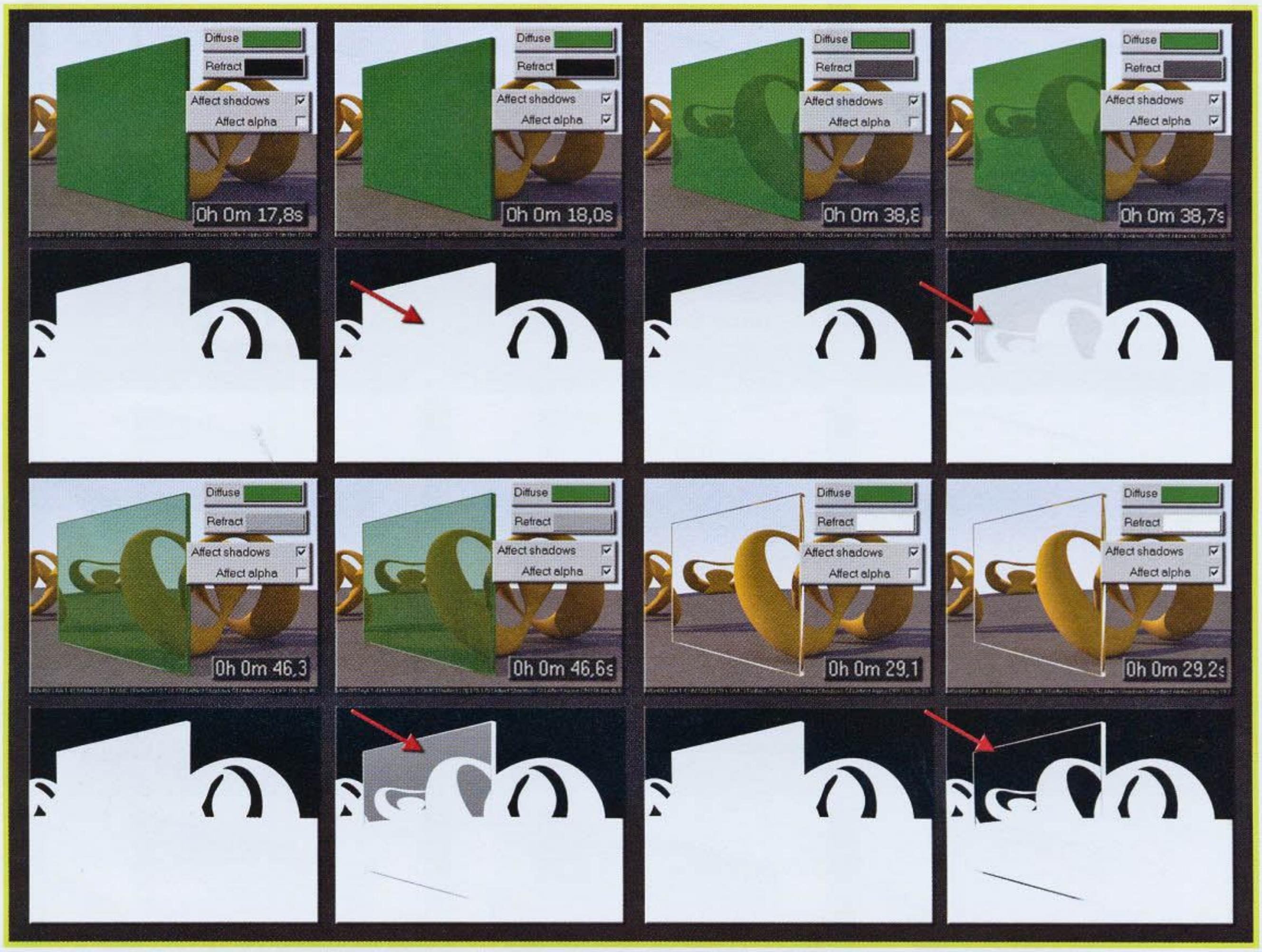


■ **Figure 5.58**
The color of refractions influences the color of the shadow produced by the glass wall. In order to have a neutral shadow one must set refraction color to a grey hue.

Moreover, also color intensity influences the intensity of shadows.

■ **Figure 5.59**
Notice how neutral colors generate shadows without color.

Affect alpha - by activating this parameter, when there are transparent objects, the ***alpha*** channel takes refraction into account. For now this option works only with objects without ***glossy*** effects. For ***glossy*** objects the ***alpha*** channel appears solid.



■ **Figure 5.60**
Effects of the ***Affect alpha*** option.

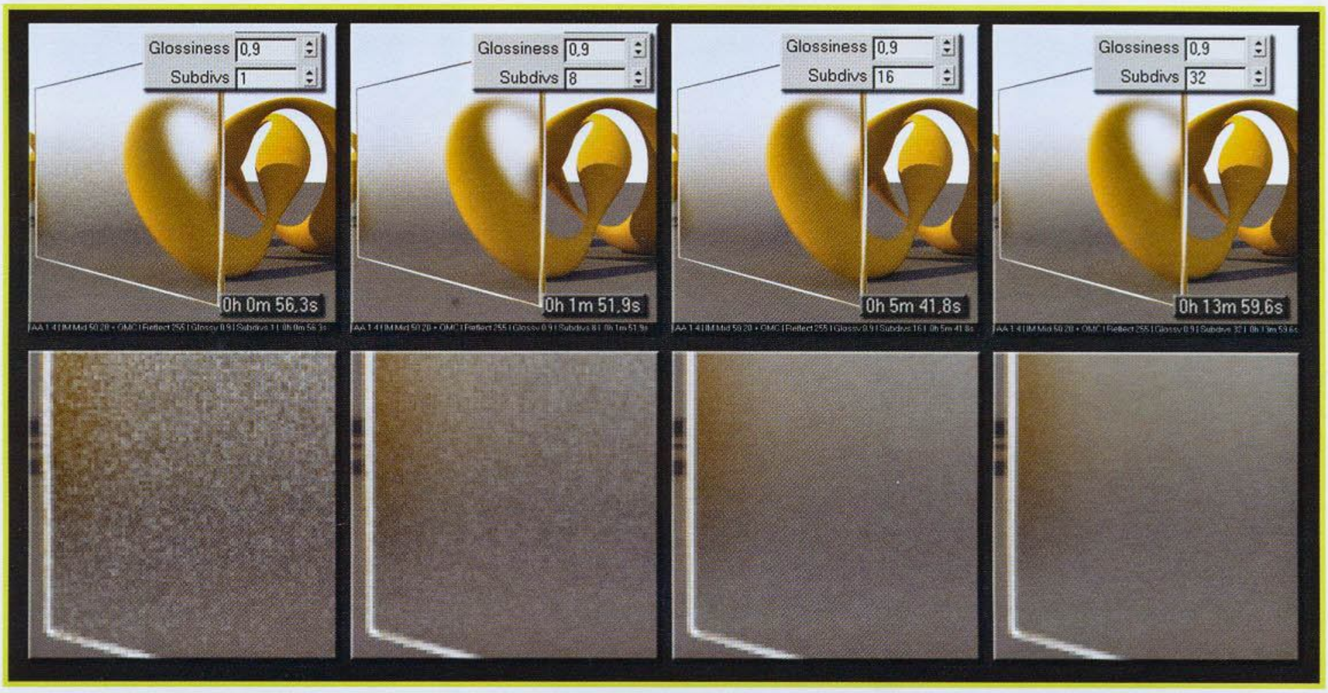
The activity of the ***Affect alpha*** parameter can be seen only in the ***alpha*** channel of the render. In these images the transparent object does not have the usual white color, typical of solid objects, but is colored in shades of grey. This feature is useful for post-production applications.

Glossiness - this controls the amount of refraction blurring. Value 1.0 indicates perfect refraction, typical of transparent window panes. Values below 1.0 start to blur refractions allowing frost effects and so on.

■ **Figure 5.61**
Frosted glass: an effect that can be reproduced by using the ***Glossiness*** parameter. With ***Glossiness*** set to 0,9 rendering time has already doubled. It is easily understandable how effects like ***glossy***, both for reflections and refractions, are very demanding in terms of hardware resources.

In this example there are two typical traits of ***Glossiness***: its amount and its rendering time. These two factors are directly proportional. As the effect is increased, so does rendering time. Also, as ***Glossiness*** is increased, its blurring concerns objects near the plane more and more. In actual fact there is another element for ***glossy*** effects: its quality which depends on ***Subdivs***. Like reflection, also for refractions one can use a map for defining ***glossy*** effects.

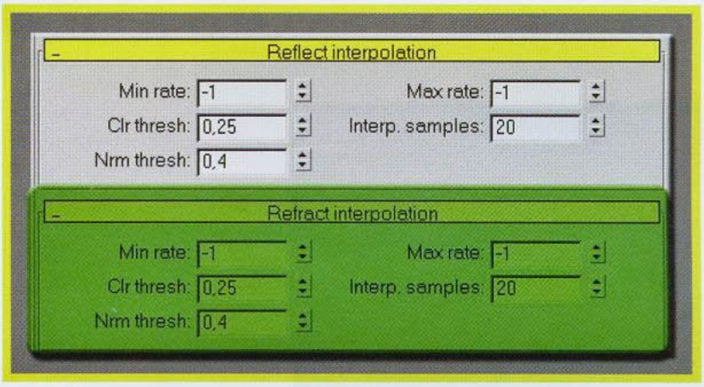
Subdivs - this controls the quality of ***glossy*** refractions. High values mean better images, at the cost of time demand. Low values quicken the process but produce refractions of scarce quality and with a lot of noise.



■ **Figure 5.62**
The quality of frosted glass depends on the number of ***Subdivisions*** used.

The ***glossy*** effect is demanding. For high quality one must typically use ***Subdivision*** settings between 8 and 16. Rendering time consequently goes up significantly. By using ***Subdivs*** = 32, one reaches 14 min., but the quality is no payback for the amount of time needed.

Just like in the case of reflections, also for refractions one can use interpolation and the parameters are exactly the same ones as those found for reflections.



■ **Figure 5.63**
The two rollouts, ***Reflect interpolation*** and ***Refract interpolation*** are identical.

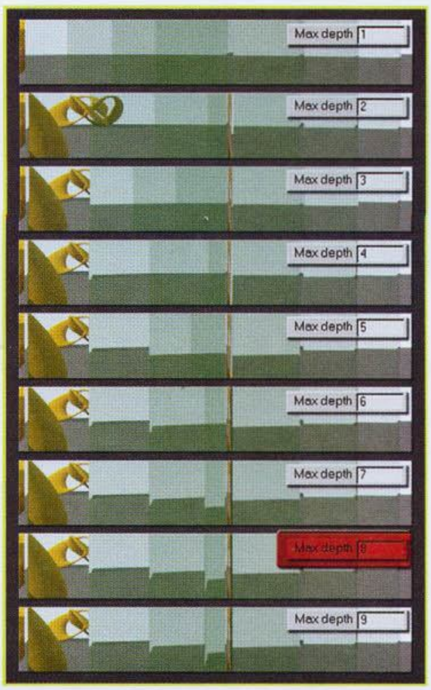
These parameters are not explained as they are identical to those covered for reflection which we have seen.

Max depth - it represents the number of times rays can be refracted. For a correct render, scenes with lots of objects with refractive properties require high settings.

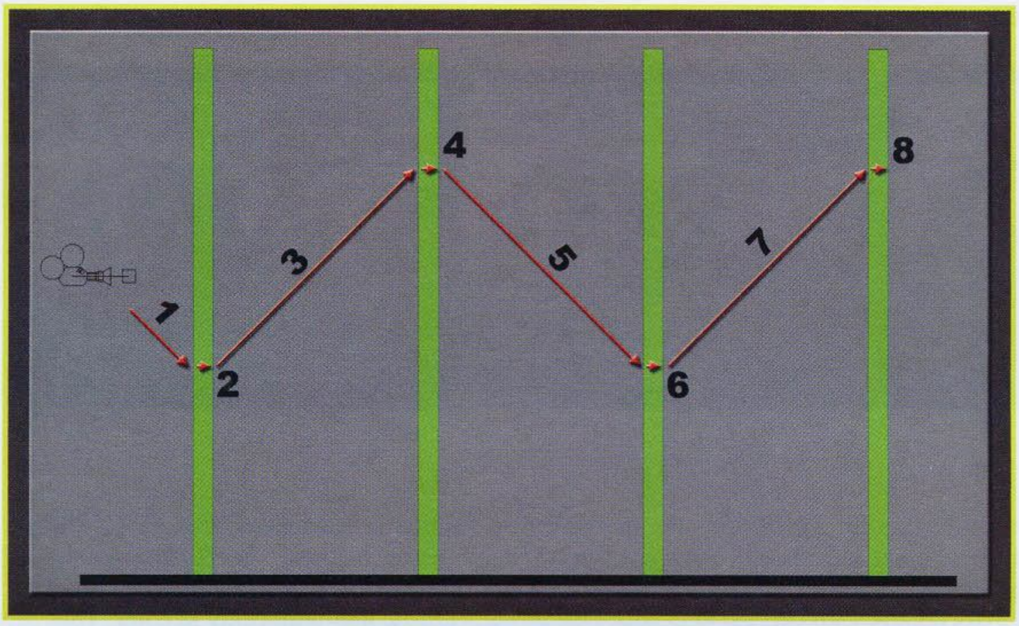
■ **Figure 5.64**
Increase in refraction quality produced by the ***Max depth*** parameter.

When there are overlapping transparent objects, for a correct refraction, the ***Max depth*** parameter must be increased. This way rays have a greater depth and can bounce more times. In the example this phenomenon can be seen mostly on the line of the horizon.

■ **Figure 5.65**
A detail. One can see how beyond ***Max depth*** = 8 no variations occur. This is because the number of rays is enough to cover all refractions. If the number of refractions was not reached, the object or objects are rendered as transparent but without refraction. That is, they do not have the ability to deflect light rays. If one observes the first image with ***Max depth*** = 1, the shadow generated by the yellow object on the far left is perfect, whereas in the last image with ***Max depth*** = 9 it is subject to refraction and appears segmented.

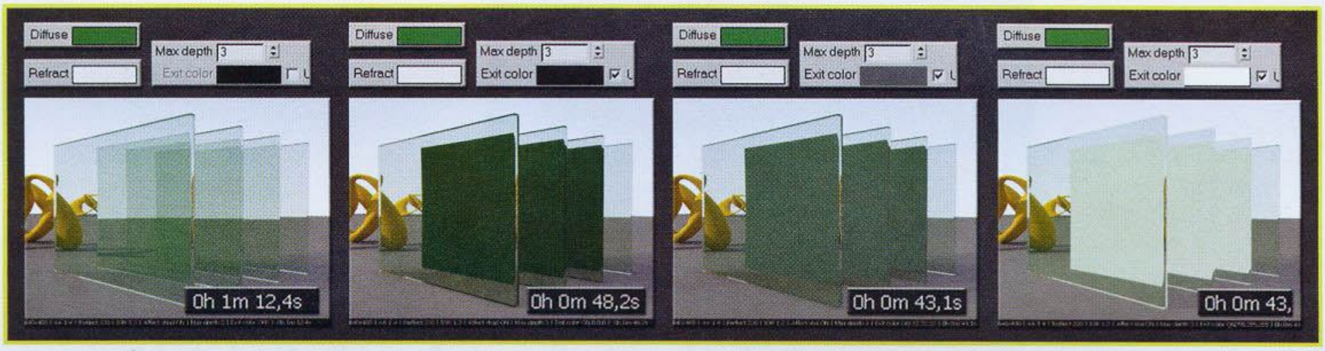


If we analyze the following image, we can see why, in the previous example, value 8 is the one that permits a correct reproduction of refractions .



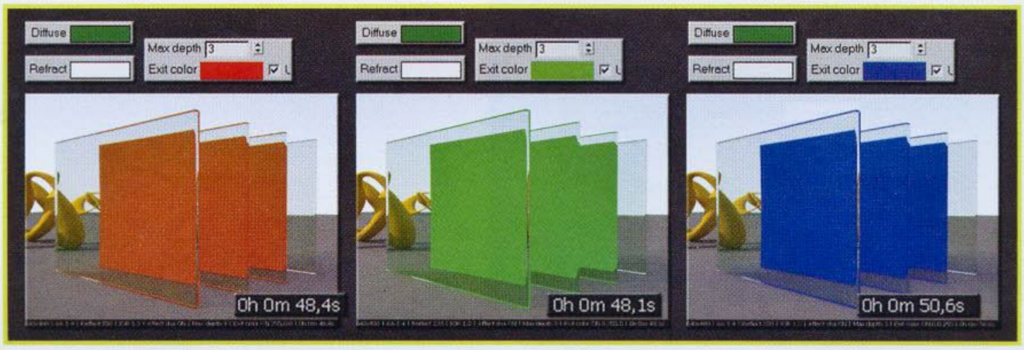
■ **Figure 5.66**
Side view of the four panels. By observing the trend of the curves it is easily understandable why ***Max depth*** is set to 8. The incoming ray must be deviated 8 times in a row in order to get through all the panels.

Exit color - when refraction rays reach the number of bounces set by the ***Max depth*** parameter, the geometry uses this color for coloring objects. This way the object is no longer transparent though.



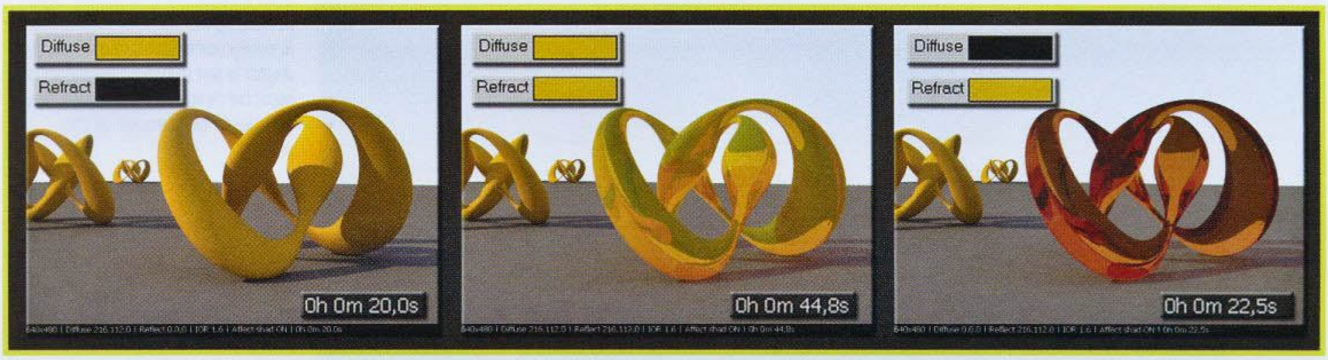
■ **Figure 5.67**
Coloring of objects which have not been reached by refraction.

In order to make an example of this parameter, a very low ***Max depth*** setting has been used. This way walls are not completely reached by refraction and one can therefore observer the effects of the ***Max depth*** parameter.
In the first image, walls not reached by refraction are transparent, like in the previous examples. By activating ***Exit color*** clicking on the checker, ***VRay*** uses a color for those parts of the object which are not refracting. If one is using a range of colors from black to white, ***VRay*** colors these areas with colors which go from green to white. This is because the ***Diffuse*** color of walls is set to green. By using different colors, here are the results.



■ **Figure 5.68**
Exit color set with different colors.

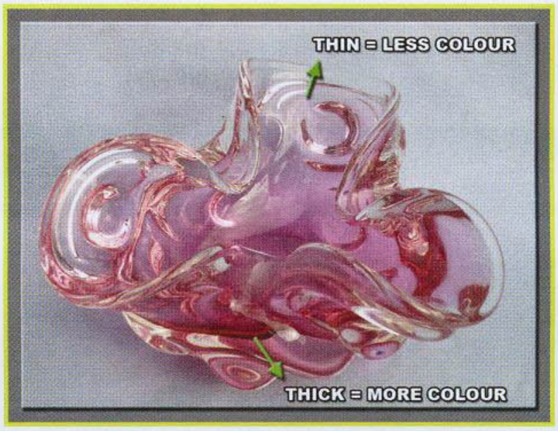
Fog color - this parameter regulates the attenuation of light as it travels through the object. It allows one to simulate the physical property of glass, thanks to which thin geometrical objects are more transparent and less colored than thicker ones. Before we look into the explanation of this parameter, we must understand some significant differences and relationships between ***Diffuse*** color and refraction color.



■ Figure 5.69
Different results in refraction.

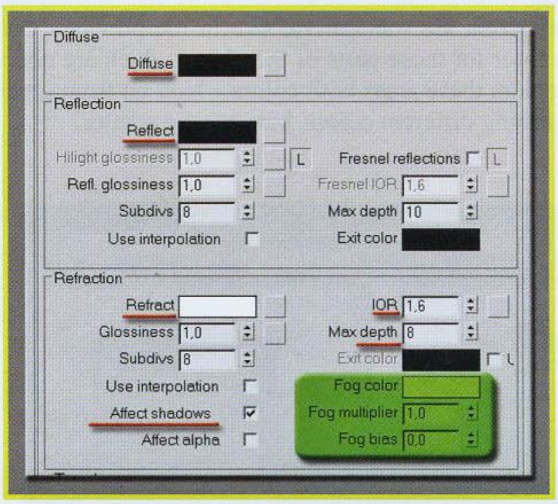
In the first image there is no refraction. In the second one the color of refraction is identical to that of the ***Diffuse*** channel. The material is semi-transparent and colored, but is not very similar to glassy material. The ***Diffuse*** component does not allow a correct coloring. Instead if we change ***Diffuse*** color to black, the object appears glassy.
As we can see in the photos, glass objects with varying thickness have a characteristic non-uniformity in color. They are not all the same color like the object in the previous test, which is instead.

■ Figure 5.70
Real object made of glass. The thinner parts are less colored than the thicker areas.

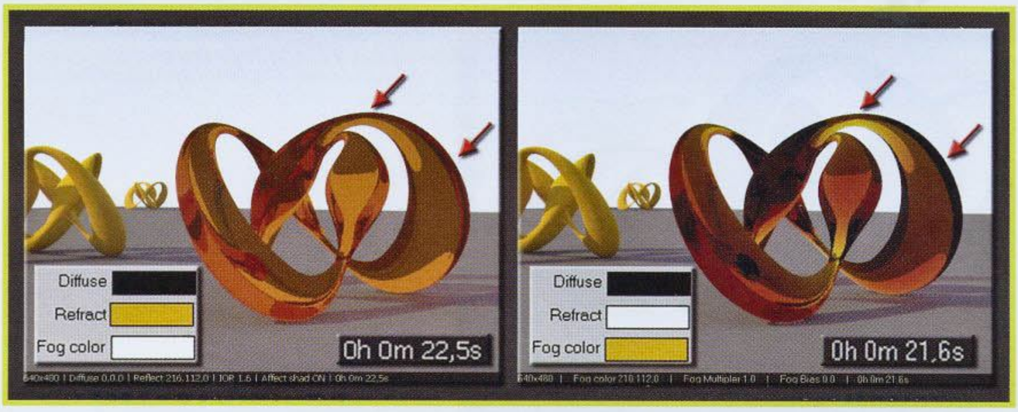


To simulate this particular effect it is necessary to adjust ***Fog color***, ***Fog multiplier*** and ***Fog bias***. In the following tests we will use underlined parameters. Only parameters concerning ***fog*** will be modified. With the current settings one obtains a completely transparent glass, without reflections. This makes it easier to analyze and understand the effects of the ***Fog*** parameter.

■ Figure 5.71
The parameters which have been changed and will be left unaltered for the rest of the tests regarding ***Fog*** have been underlined in red. The three ***Fog*** settings are shown in the green box.

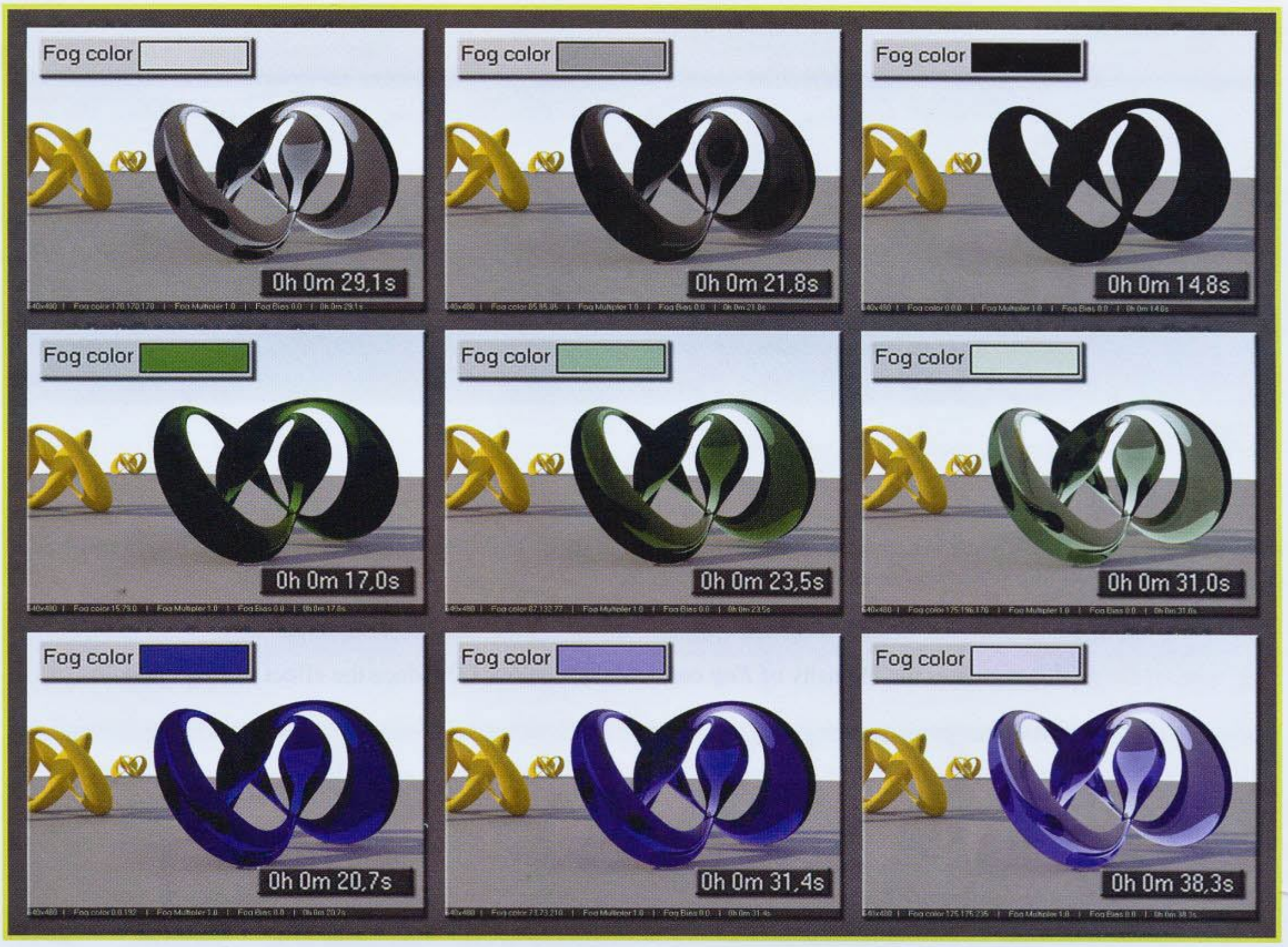


The following image shows how a render can change radically when one alters ***Refraction*** and ***Fog*** channels.



■ **Figure 5.72**
Fog color in action.

As the red arrows indicate, in thinner areas of the object in the right-hand image, the color of glass is brighter than in the thicker parts. This does not happen instead in the left-hand image, where the color in the ***Refraction*** channel does not allow this effect and the result is a uniform color in the whole object. In the following image different color tones are used, black, green and blue. The Whiteness parameter is modified in the Color Selector of 3ds Max.



■ **Figure 5.73**
Different ***Fog colors***.

Notice how the variation of colors from bright to darker ones modifies the render significantly in terms of color, brightness and transparency of the material itself; as a result also the shadow produced adapts to the new color. This is because ***Affect shadows*** is active.

Another consideration worth making is that ***Fog color*** depends completely on the size of the object and on the scene.

In the following test the object has been duplicated and its size modified, maintaining the same material: the ***Fog color*** is also the same.

■ **Figure 5.74**
The same object, duplicated and scaled with the same material applied.

The objects appear to be very different. The smaller one, being also thinner because of its size appears to be brighter. Vice versa, the larger one appears darker.

A similar thing happens when one multiplies the size of the whole scene.

■ **Figure 5.75**
The same scene, first enlarged by ten times and then reduced by ten times.

Knowing the scale one is working on is of fundamental importance. The same material used in a scene could be too dark or bright if used in a scene with a different drawing scale.

Fog multiplier - this regulates the intensity of ***Fog color***. Values below 1.0 reduce the effect of ***Fog color***.

■ **Figure 5.76**
Variation of the ***Fog multiplier*** parameter.

By changing this parameter, ***Fog color*** intensity increases or diminishes.

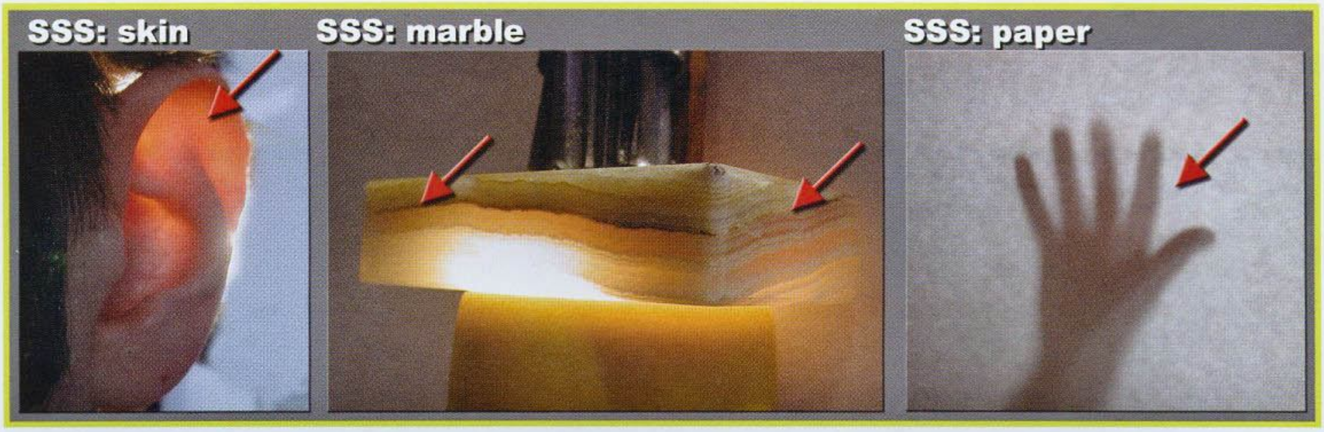
Fog bias - this parameter manages the difference of intensity of the *Fog color* effect between the thicker and thinner areas.

■ Figure 5.77
Fog bias modifies the difference in intensity between dark and bright zones.

With values greater than zero, the material tends to lose its typical *Fog color* properties. Vice versa, with negative *Fog bias* values, the difference in color, intensity and saturation increases between areas with thicker or thinner geometrical sections. Its variation, however, does not generate physically correct results.

TRANSLUCENCY or Subsurface scattering (SSS)

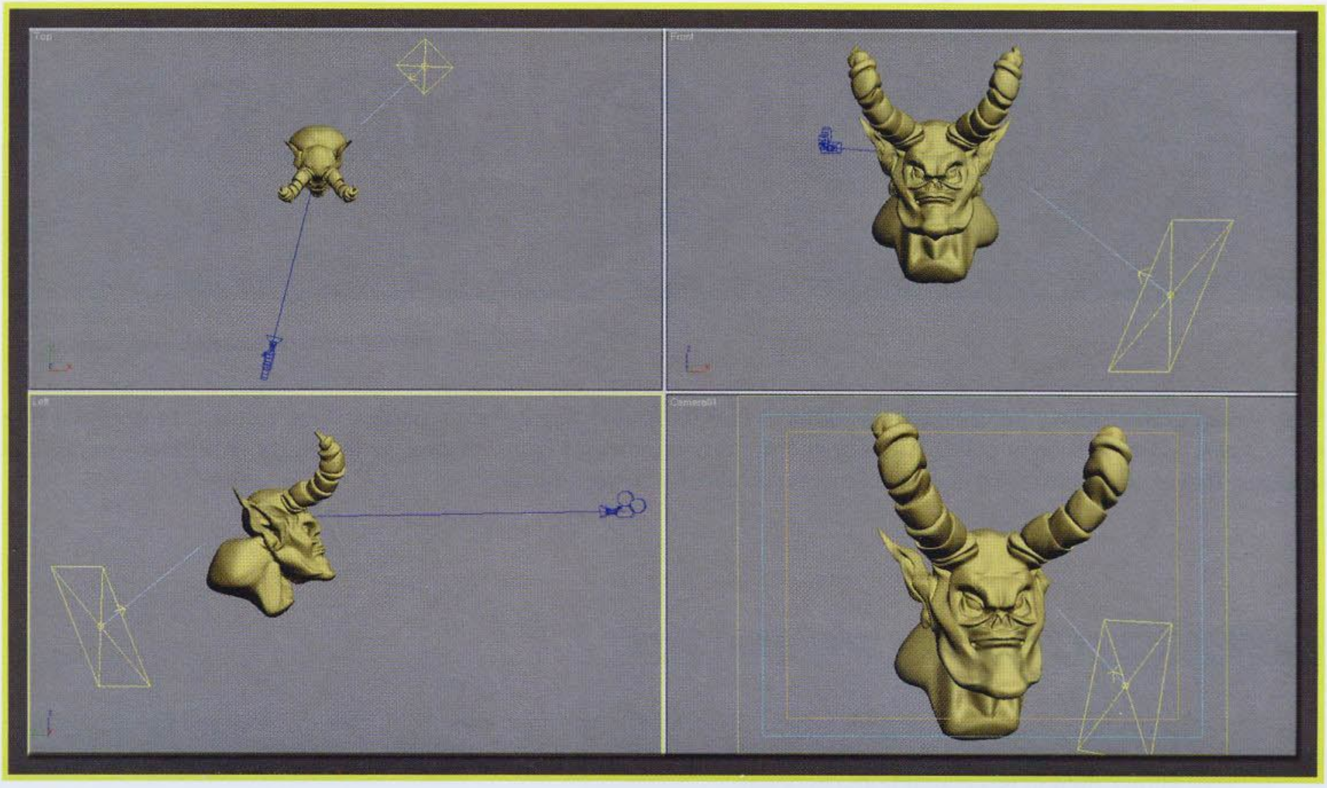
Translucence is a characteristic of materials which allow light to spread within them. One can imagine that it might be necessary for such a material to be transparent, but unlike glass or frosted glass, which only blurs refractions, translucency allows light to travel within the material.



■ Figure 5.78
Real-life examples of translucency.

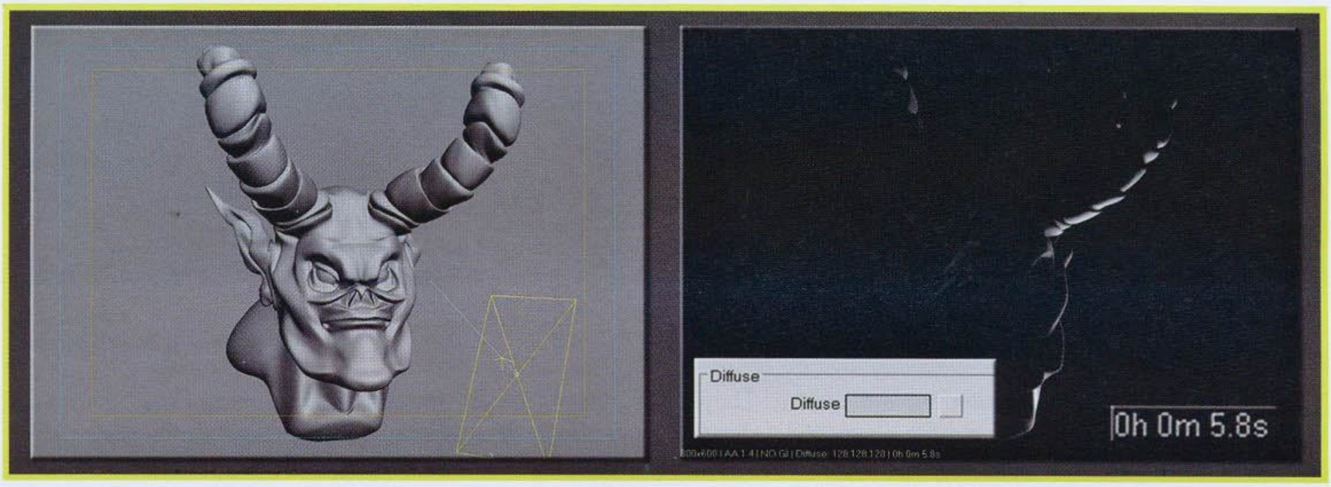
These objects are not transparent but translucent. They allow light to travel through them. Human skin, illuminated from behind, becomes reddish, marble glows and a sheet of paper allows what is behind it to be seen on the other side.

The used for the tests is made up of a 3D model illuminated by a ***VRayLight***. The whole thing rendered without GI. This way we can observe the ***SSS*** effect better.



■ **Figure 5.79**
The scene being studied.

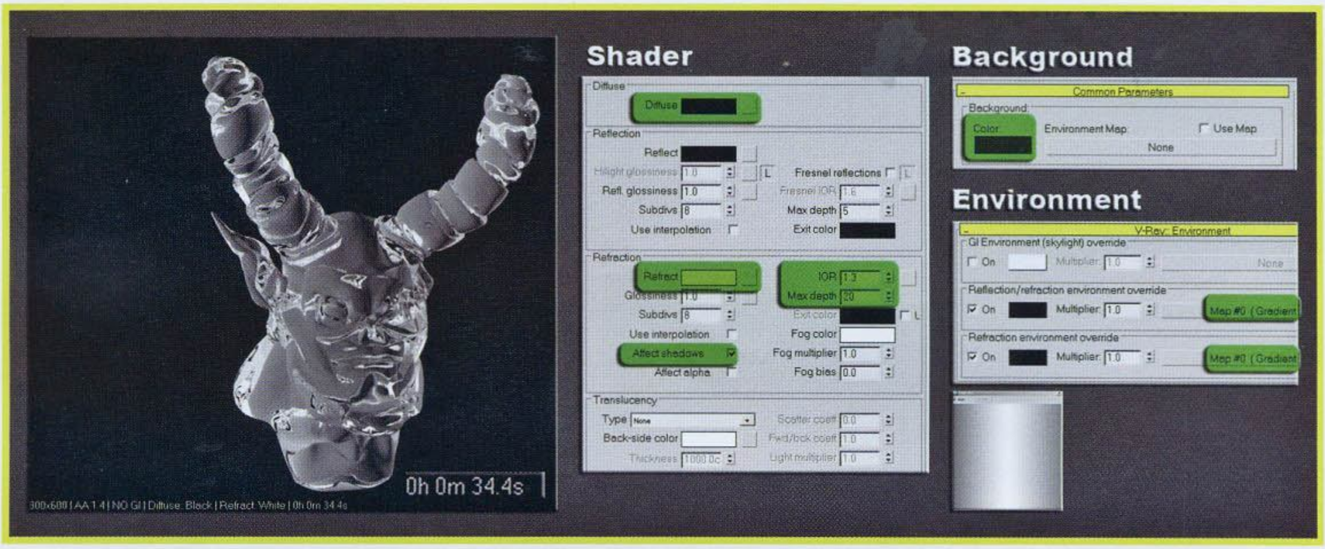
Before getting into the explanation of to create ***SSS*** materials, we will take a look at a render with an average grey color. The ***VRayLight*** is the only light source.



■ **Figure 5.80**
Grey material without any reflection, refraction or ***SSS***.

The demon has a slight illumination which allows one to be aware of its presence. Also, as there is no GI, the left part of the mesh is completely black.

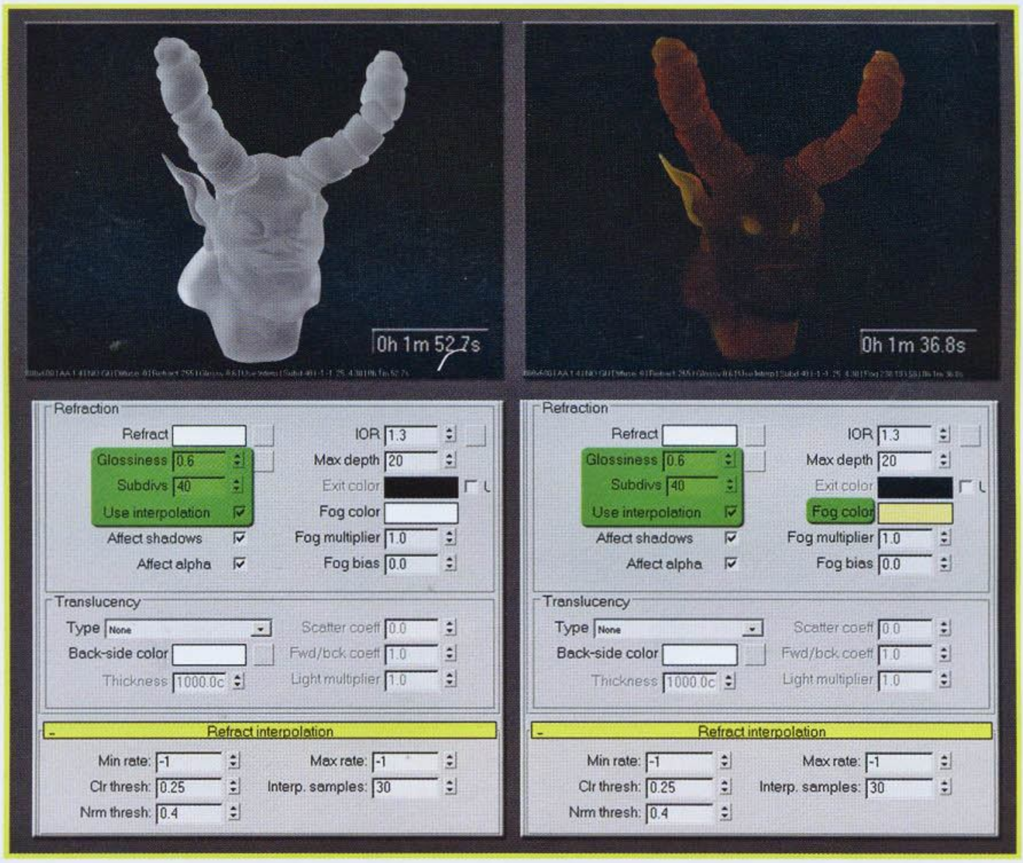
Translucence needs a certain amount of refraction to exist because the object needs to be a little transparent. We shall begin by creating a completely transparent glass, increasing refraction and setting ***Diffuse color*** to black. Also the ***IOR*** has been modified and ***Max depth*** as well. Lastly the ***Affect shadows*** parameter has been activated.



■ **Figure 5.81**
Applying refraction to the model.

As the background is completely black, in order to observe refraction, the ***Environment*** of ***VRay*** has been modified by inserting a gradient map in refraction and reflection channels. We are still far away from the desired effect though. In the meantime rendering time has increased a great deal.

The next step is to apply the ***glossy*** effect and ***Fog color***, the two subjects we have just seen a few pages ago.

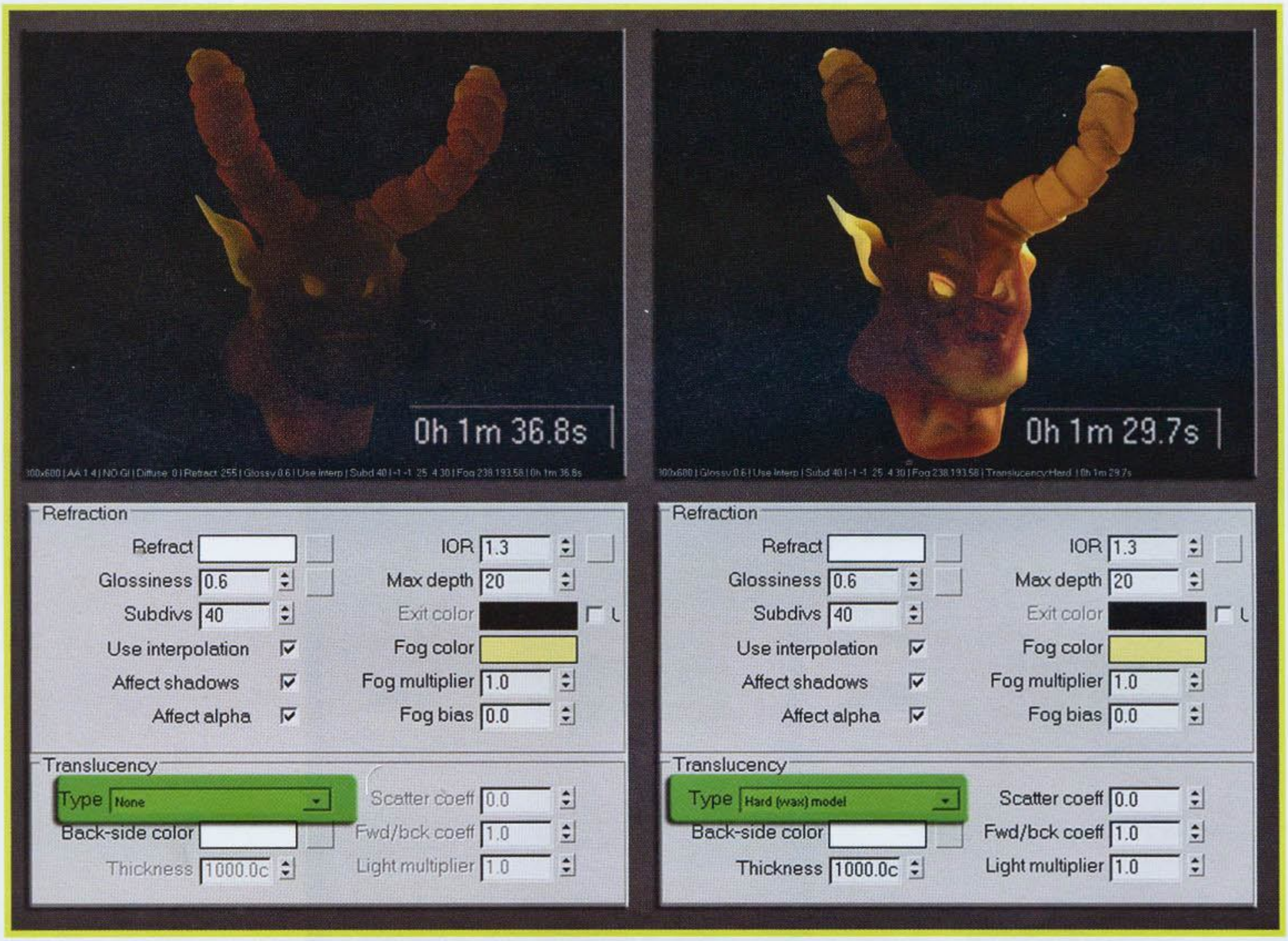


■ **Figure 5.82**
Use of refractions with interpolated ***glossy*** effect and ***Fog color***.

The ***SSS*** effect starts to be visible. Now the material is more similar to a translucent object. The ears are brighter compared to the rest of the body and so are the horns. The demon's chin, being the thickest part, is also the darkest. ***Translucency*** has still not been activated. Notice how the object is not in the least bit affected by the light source: if we turn the ***VRayLight*** off the result does not change.

■ **Figure 5.83**
Without translucence, the model does not take the ***VRayLight*** into account. Also, without a light source rendering time has been lowered. The "luminosity", if we can call it so, is due to the blurred refraction of the Environment.

By activating translucency and by using default values, the result changes completely.

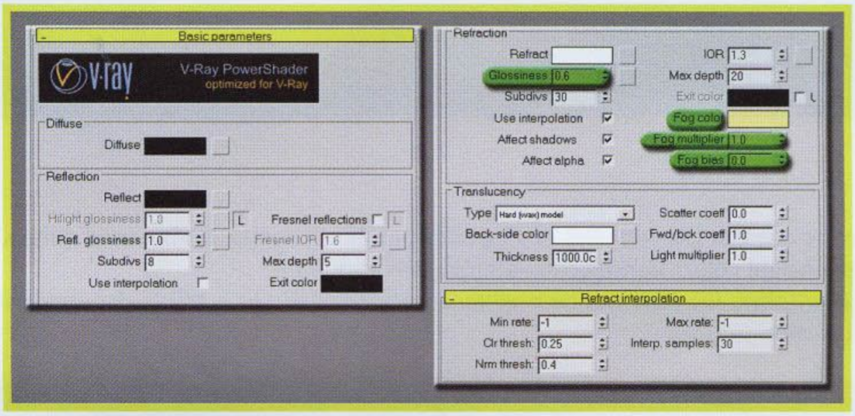


■ **Figure 5.84**
Activation of translucency.

Now if we observe the model, we notice that the right side of the demon's head, where the ***VRayLight*** is placed, is lit up more and also the horns. Also on the chin and on the back of the neck one can see the light which has traveled through from the right side of the head. By turning the ***VRayLight*** off the model is completely dark. Rendering time has not gone up at all and in merely 90 seconds we have obtained a good quality render.

■ **Figure 5.85**
Deactivation of light sources during the use of ***SSS***: materials appear completely dark. One should remember that for the test regarding the ***SSS*** no GI is used.

As can be easily noticed, for a correct translucency result some fundamentally important factors are the settings in the ***Refraction*** channel and in particular the amount of ***Glossiness*** and ***Fog color***. Before analyzing specific ***SSS*** parameters, one must understand how the ***Refract*** channel acts upon translucency. First of all we will analyze the behavior of ***Glossiness***. The parameters used for the material are the ones shown in the image. The values which influence the ***SSS*** the most, which will be submitted to an in-depth analysis are those highlighted in green.



■ **Figure 5.86**
The 4 main parameters for ***SSS*** definition.

When in the presence of ***SSS***, by modifying the setting of ***Glossiness*** of the ***Refraction*** channel, the result is the following:

■ **Figure 5.87**
Change in ***Glossiness*** referred to translucency.

With ***Glossiness*** = 1.0, an incorrect setting if one wishes to reproduce a translucency effect, rendering times are very high and it is certainly a setting not to be used. As intensity lowers, one can make two considerations. The first regarding rendering time which slowly diminishes. The second, more important, concerns the level of detail which tends to disappear. This is because the ***glossy*** effect is an effect which blurs refractions. The more they are blurred, the less detail there is. This explains the lower rendering time. It is advisable, for ***SSS*** effects, to use values between 0.1 and 0.5.
Another very important parameter for translucency definition is ***Fog color***.

■ **Figure 5.88**
Fog color and translucence.

In order to determine ***SSS*** color one can use the color of ***Fog color***. In this example, having used quite a high setting for ***Fog multiplier***, also with very bright colors translucency saturation is very high.
Instead by modifying ***Fog multiplier***, the ***SSS*** can be brightened or darkened. As a result also rendering time suffers: with "less" ***SSS*** the render is faster.

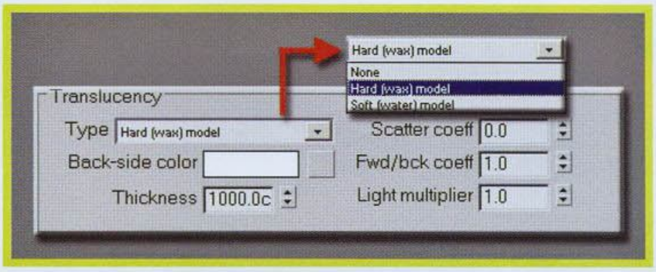
■ **Figure 5.89**
The ***Fog multiplier*** parameter and translucence.

Lastly the ***Fog bias*** parameter.

■ **Figure 5.90**
The ***Fog bias*** parameter and translucency

With this last parameter one can vary the intensity of ***SSS*** between thicker and thinner areas. With ***Fog bias*** = 2.0 the model is subject to ***SSS*** in a uniform way, regardless of geometrical thickness. With ***Fog bias*** = -1 geometry has a high contrast, with high ***SSS*** in the area of the ears and not much transparency on the rest of the face.

It is time now to analyze the last parameters of the ***Basic parameters*** rollout, those concerning actual translucency.



■ **Figure 5.91**
Translucency management.

Type - this pull-down menu allows one to select one of three calculation modes of the ***SSS***.

None - no translucency.
Hard model - this activates ***SSS*** with shiny translucency effects.
Soft model - activates ***SSS*** with moderate translucency effects which have less contrast compared to the ***Hard*** method.

■ **Figure 5.92**
The three modes for translucency made available in VRay.
--- DVD ---

In the first box on the left, if translucency is turned off, the only effect produced is simply due to blurred refraction as well as the color blends caused by ***Fog color***. By activating ***SSS***, the material transports light energy and by using the ***Hard*** mode, this effect is more evident than when one uses the ***Soft*** mode, which has smoother and more uniform color blends.

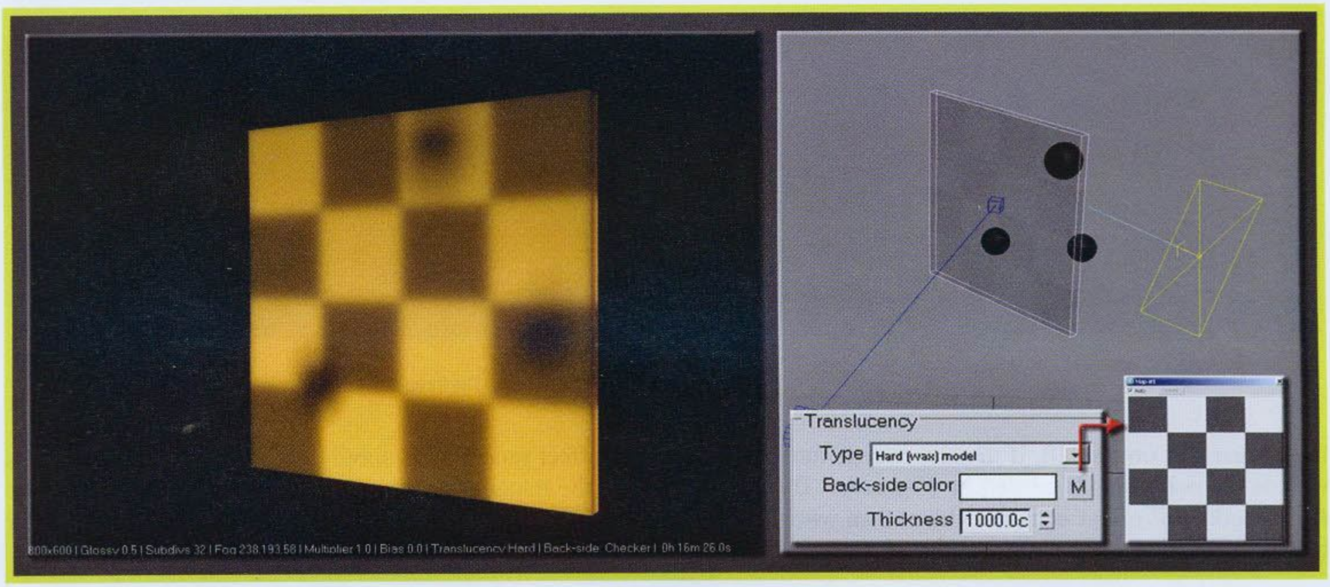
Back-side color - by using the color offered by this parameter, one can manage the amount of translucency in the material, similarly to refraction and reflection channels.

■ **Figure 5.93**
Thanks to the color found in the ***Back-side color*** one can define the quantity of ***SSS*** found in the model.

It is also possible to combine ***Fog color***, with all its parameters, together with ***Back-side color***, in order to obtain interesting effects.

■ **Figure 5.94**
Combined effect of ***Fog color*** and ***Back-side color***.

Lastly, in order to define ***SSS*** position and intensity one can use any map, just like one can do for ***Diffuse***, ***Reflection*** and ***Refraction*** channels.



■ **Figure 5.95**
A procedural 2D map being used for ***SSS*** definition.

Thickness - this setting limits the depth of calculation of the ***SSS*** within geometries. Useful if one does not need to calculate ***SSS*** for the whole volume of the model, but only on the outermost areas.

■ **Figure 5.96**
Variation of the ***Thickness*** parameter, a method for limiting the ***SSS*** to the outer surface alone.

The model used in the test is 50 cm wide. It is therefore possible to appreciate the effect of the ***Thickness*** parameter only at this size. What can be seen is a gradual decrease in ***SSS***, until it disappears completely. Obviously rendering time decreases with ***Thickness***.

Light multipler - multiplication of the ***SSS*** effect.

■ **Figure 5.97**
Variation of the ***Light multiplier*** parameter.

It is a very simple parameter which allows one to multiply, from within the model, the intensity of light coming from the source without altering the ***VRayLight*** directly.

Scatter coefficient - this value indicates the amount of distribution of light within the object. With a setting of 0.0 rays are distributed in all directions. With a value of 1.0, rays never change direction.

■ **Figure 5.98**
Variation of the ***Scatter coeff*** parameter applied to both an organic model and a box.
- - - DVD - - -

In the organic model, even setting minimum and then maximum values, the difference between the two tests is slight. In the box instead, the variation of the ***Scatter coeff*** leads to a very evident difference on the right side, parallel to the rays of light. With ***Scatter coeff*** = 1 light spreads in a brighter fashion, although the back face, which is the one seen up front in the image, has the same degree of intensity in both cases.

Foward/backward coefficient - this controls the direction of rays for ***SSS***. Set to 0.0 means that rays can only go forwards within the object; 0.5 means that rays have the same possibility of propagating forward and backward within the material. With a setting of 1.0 the ray is distributed only backwards.

■ **Figure 5.99**
Variation of the *Forward/backward coefficient*.
--- DVD ---

With ***Scatter coeff*** = 1, renders have been carried out with different ***Fwd/bck coeff***. Therefore one obtains a greater brightness in shadowy areas, the areas lit by a transposition of light and not directly illuminated by the ***VRayLight***. Its effect is visible both in the box and in the organic model.

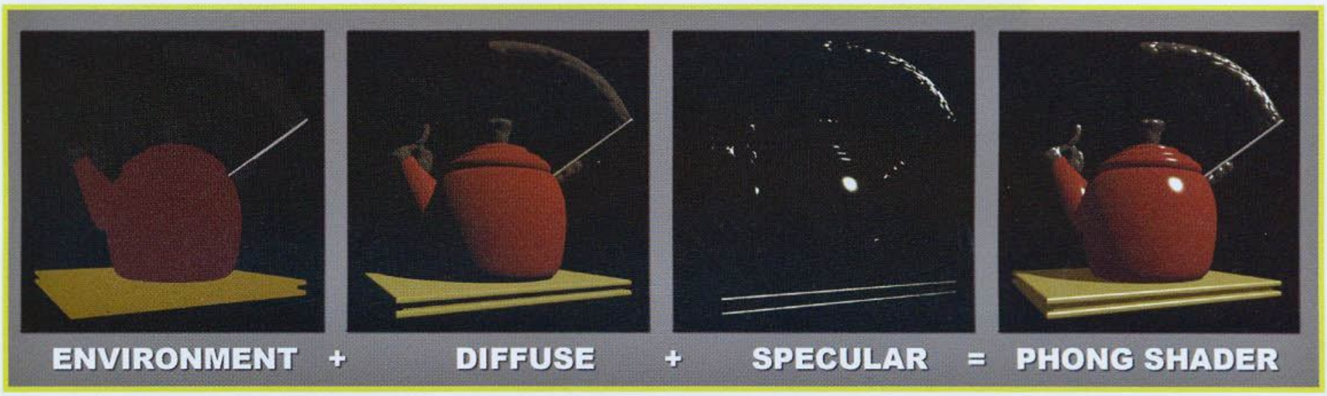
BRDF

In the early age of computer graphics, the simplest illumination models considered only the portion of light which travels from a source to an object: this was named direct illumination.
The way light is reflected by an object can be described by a mathematical formula, called **B**idirectional **R**eflectance **D**istribution **F**unction or ***BRDF***.

$$f_r(x, \omega_i \rightarrow \omega_r) \equiv \frac{L(x, \omega_r)}{L_i(x, \omega_i)\ \cos\theta_i\, d\omega}$$

■ **Figure 5.100**
BRDF equation.

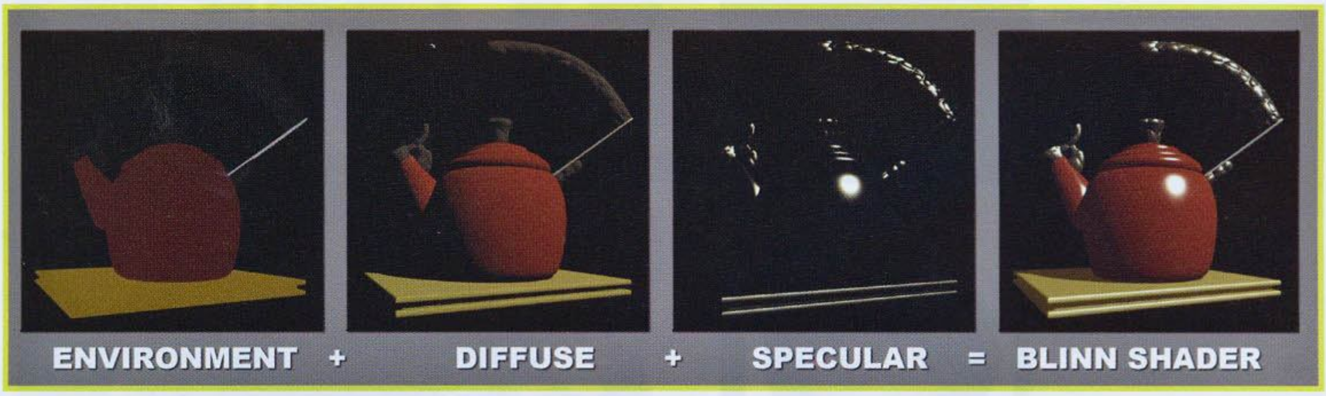
The first rendering systems used a simplification of this formula and would calculated direct illumination as a sum of two elements: the diffuse part and the specular part. The diffuse part, also known as lambertian, consisted of the part of light which was bounced off the object in all directions, instead the specular part refers to light when it is reflected on the surface of an object, like it does on a mirror.
Later Phong's reflection model was introduced, adding a third component, environmental light, which offered a basic simulation of indirect illumination.



■ **Figure 5.101**
Decomposition of a render in the three main channels: environmental, diffuse and specular light.

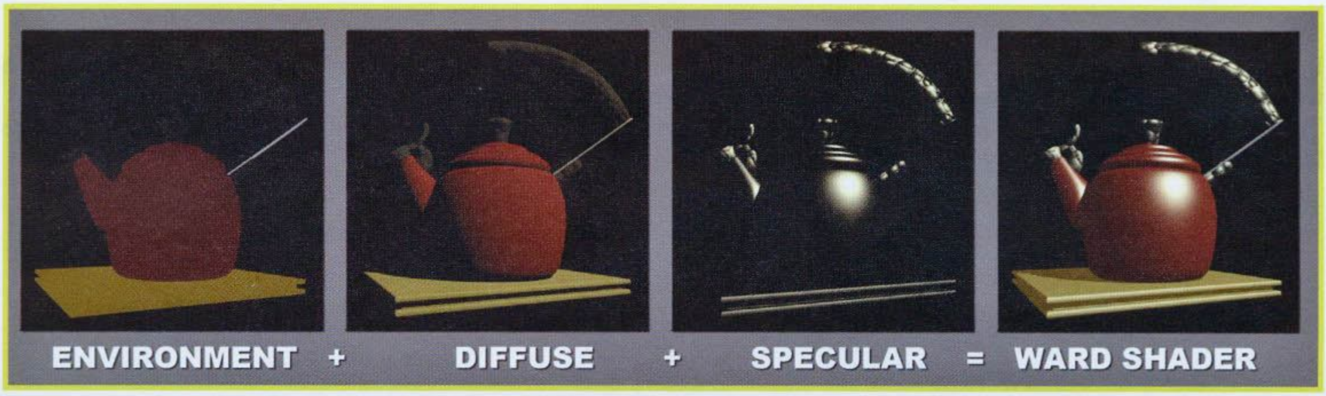
This method was invented in 1973 by Vietnamese Bui Tuong Phong, at the university of Utah. Phong shading smoothens the edges between faces and renders the areas of maximum illumination in a realistic way for regular shiny surfaces. This shader carries out interpolation of the intensities on a face according to the average number of normals on the joining faces and calculates the normal for each pixel of the face. A few years later the Blinn model was introduced.

Blinn shading is a slight variation of ***Phong*** shading. The most evident difference is in specular illumination points which appear more blurred.



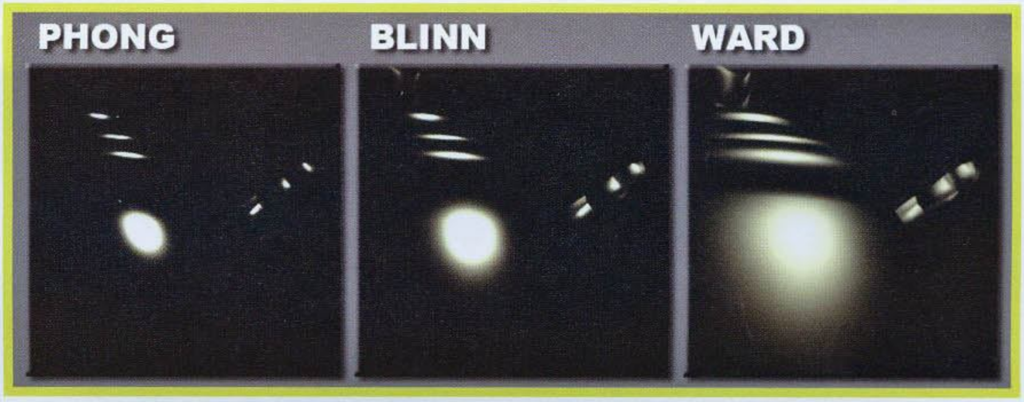
■ **Figure 5.102**
Blinn is a variation from ***Phong shader***.

VRay offers another type of shader called ***Ward***, which produces ***Highlights*** which are even more blurred compared to the ***Blinn*** method, and it is the first-choice shader for anisotropic effects.



■ **Figure 5.103**
The third type of ***Highlights*** available in VRay: ***Ward***.

■ **Figure 5.104**
The three types compared.



The parameters in this rollout allow one to obtain and simulate various types of ***Highlights***, which range from more pronounced ones typical of shiny plastic, to anisotropic characteristics, which allows one to create surfaces with the points of maximum illumination being elliptic. These points are useful for modeling hair, glass or frosted metal.

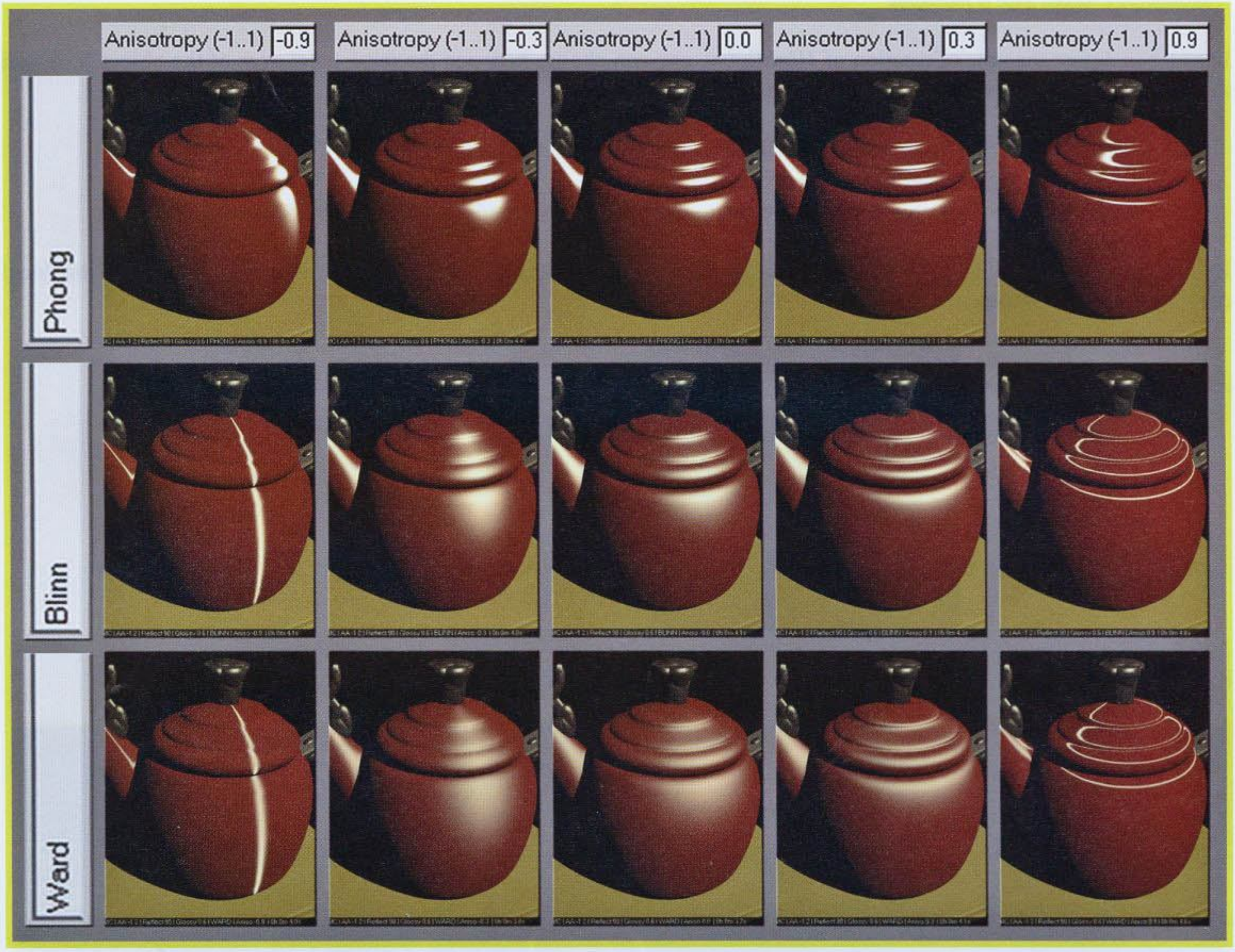
■ **Figure 5.105**
Anisotropy is the characteristic which allows ***Highlights*** to form, which are non-circular but stretched. It is a typical effect in frosted materials or reflections in hair, where highlights cannot be round because of the geometric nature of the object.

Type - this menu determines the type of ***BRDF***, in other words the shape of ***Highlights***.

- ***Phong*** - sets Phong type ***Highlights***.
- ***Blinn*** - sets Blinn type ***Highlights***.
- ***Ward*** - sets Ward type ***Highlights***.

Highlights are generated by blurred reflections obtained by decreasing ***Glossiness***. They can be physically correct in which case reflections are generated by the *VRay* raytracer, or they can be generated in a physically incorrect way. In this case blurred reflections are not visible; only spots of maximum illumination are.

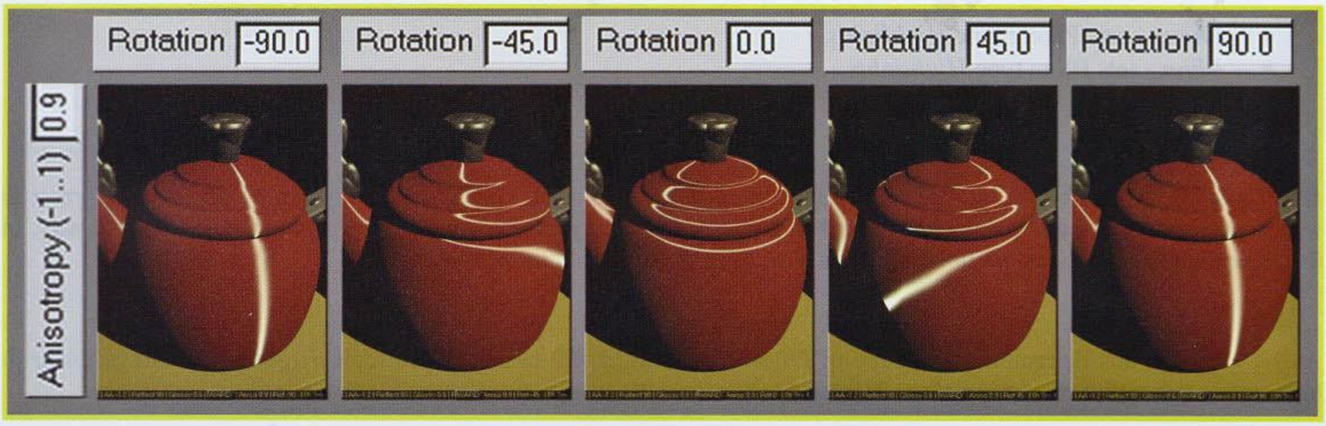
Anisotropy - this determines the shape of ***Highlights***. Setting 0.0 means an isotropic ***Highlight***. Settings lower or higher than 0.0 modify the shape of the ***Highlight***.



■ **Figure 5.106**
Anisotropy being applied by using the three shaders available.

Thanks to the ***Anisotropy*** parameter one can change the shape of ***Highlights*** from circular to elliptic, as well as size. There is no unique model for all materials, although observing reality we can see that anisotropic effects are usually very blurred and this is why it is preferable to use the ***Ward*** method. Also the ***Blinn*** produces quite smooth ***Highlights***. With appropriate ***Glossiness*** settings one can in any case use the ***Phong*** method and obtain results which are quite accurate. In the tests we have just carried out, negative values produce incorrect results for the teapot, which actually has horizontal ***Highlights*** and not vertical ones.

Rotation - this parameter defines the angle of rotation of ***Highlights***.



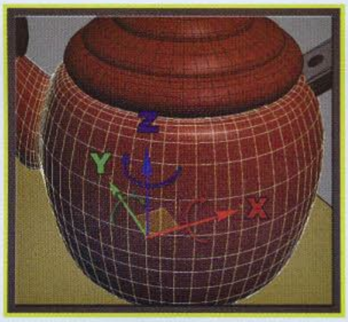
■ **Figure 5.107**
Variation of the rotation angle of ***Highlights***.

In order to define the axis of rotation of ***Highlights***, one can choose between the three local axes of the object X, Y and Z. This way Highlight rotation can be managed in every aspect.

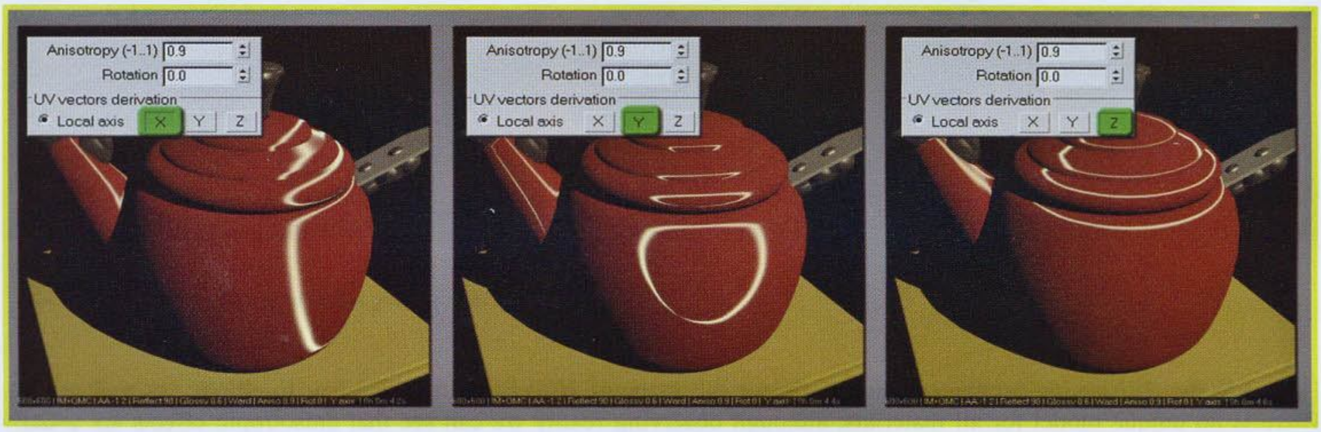
- ***Local axis*** - allows one to change the direction of the anisotropic effect.
- ***Local axis X,Y,Z*** - specifies rotation axis.
- ***Map channel*** - with this parameter activated, the direction is defined by the ***Map channel***.

By setting coordinates in local mode, one can observe the transformation Gizmo moving into line with the local coordinates of the object.

■ **Figure 5.108**
Local coordinates of the mesh with rotations on the X,Y and Z coordinates highlighted.



By default the Z axis is set with ***Rotation*** = 0. This means that ***Highlights*** are horizontally laid out.



■ **Figure 5.109**
The three directions of ***Highlights*** along the local X, Y and Z coordinates.

In this particular case, the Z axis is the one which produces correct results. ***Highlights*** along the local X and Y axis are not represented in a correct fashion.

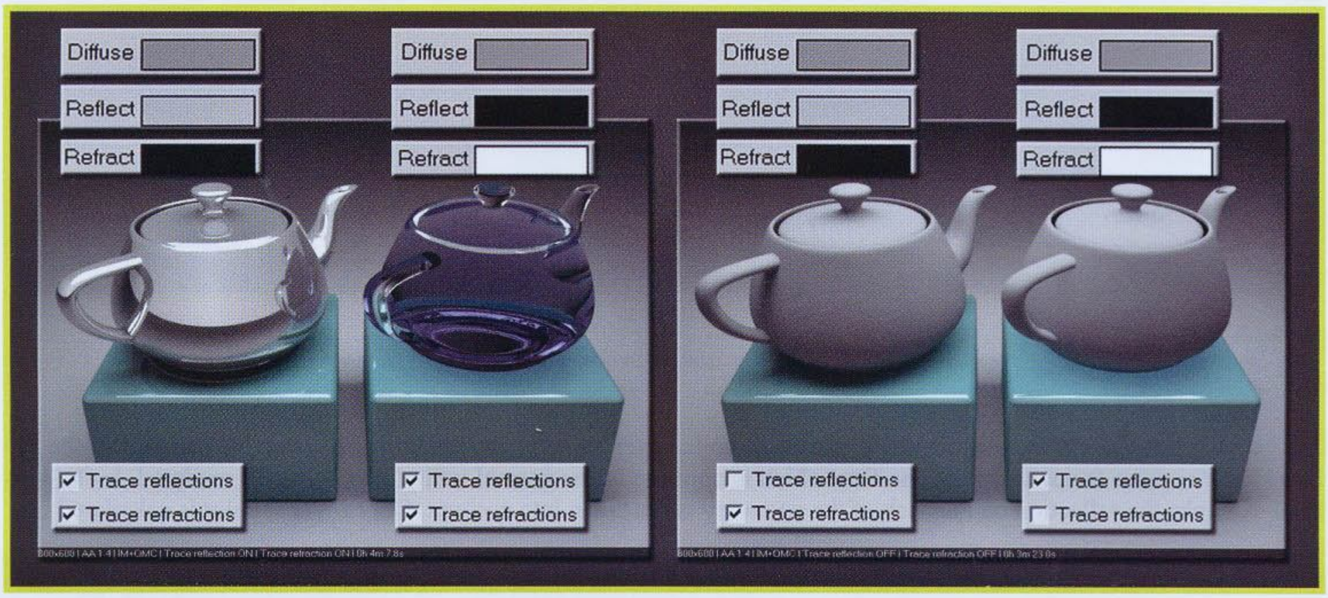
Map Channel - this offers the possibility of using map coordinates by specifying the channel.

Options

This rollout does not have a particular characteristic which distinguishes it from others. It contains general parameters which have no affinity with those analyzed so far.

Trace reflection - with the ***Trace reflection*** option deactivated, reflections are not calculated, even if ***Reflect color*** is a color different to black. This option is fundamental if one wants to create ***Highlights*** which are not physically correct or one wishes to speed up render calculation by deactivating reflections only on a specific material. ***Diffuse*** color remains unaltered and is not influenced by the intensity of reflection, exactly as if the reflection did not exist.

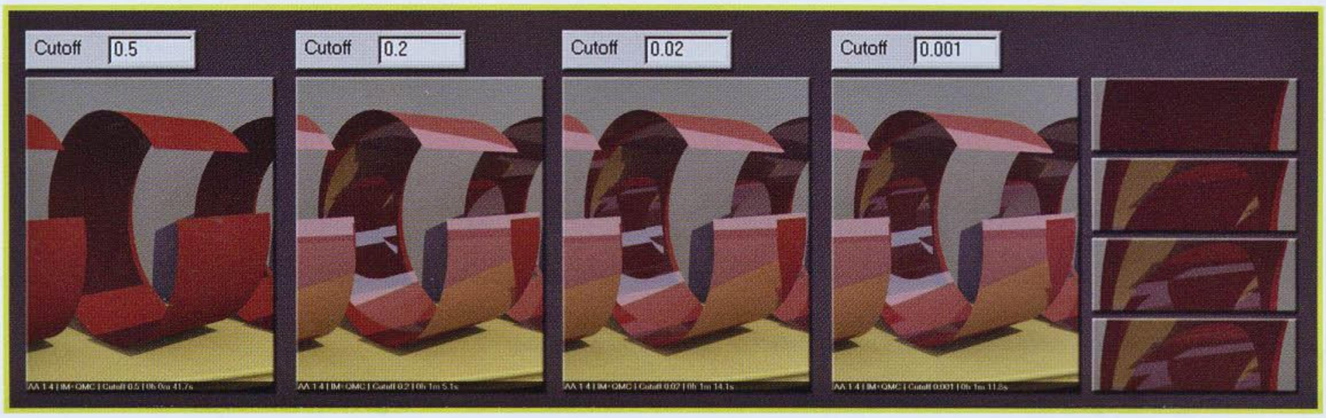
Trace refraction - when deactivated, refractions are not calculated, even if ***Refract color*** is not black.



■ **Figure 5.110**
Activation and deactivation of the calculation of reflections/refractions.

The scene is made of two models, one with reflective properties and the other with refractive ones, placed on a slightly shiny pavement. By deactivating the ***Trace reflection*** and ***Trace refraction*** options, reflection and refraction are not calculated, although the colors of these channels are different to RGB = [0, 0, 0]. In the case of the right-hand teapot, by deactivating refractions not even its shadows are produced. This is because the ***Affect shadows*** option is active and refraction is set to maximum levels.

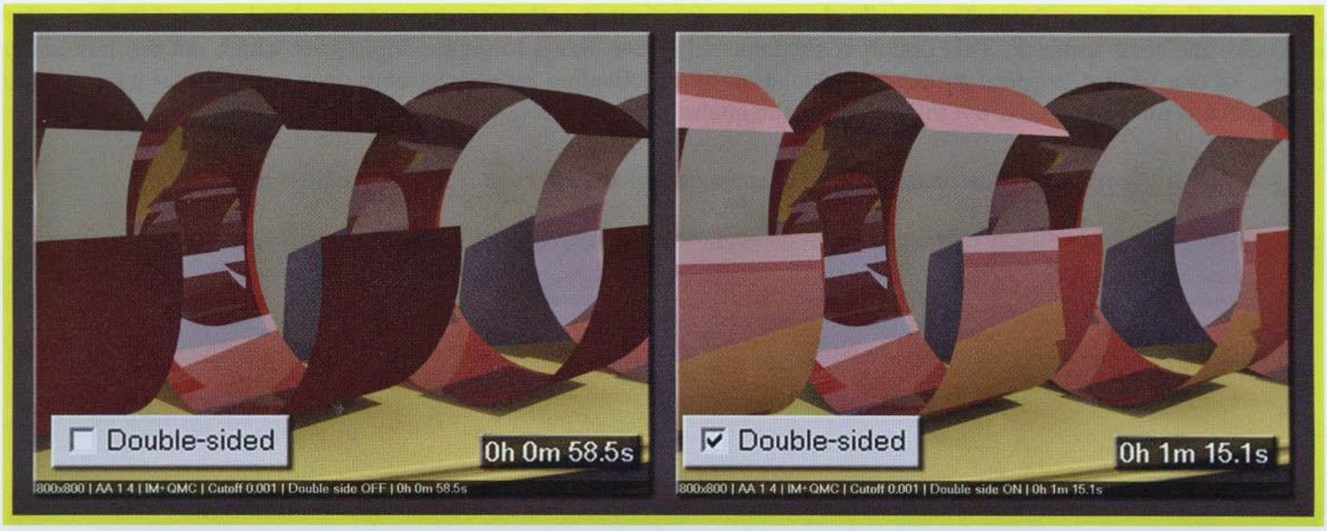
Cutoff - this represents the threshold beneath which both reflections and refractions are not calculated. A too low value, such as 0.0, can cause very long rendering times.



■ **Figure 5.111**
Reflection defects due to low ***Cutoff*** values.

Sometimes it can happen that strange halos, dark circles or problems with reflective materials occur when ***Glossiness*** or ***Fresnel*** are activated. These problems can be solved by diminishing the ***Cutoff*** setting value.

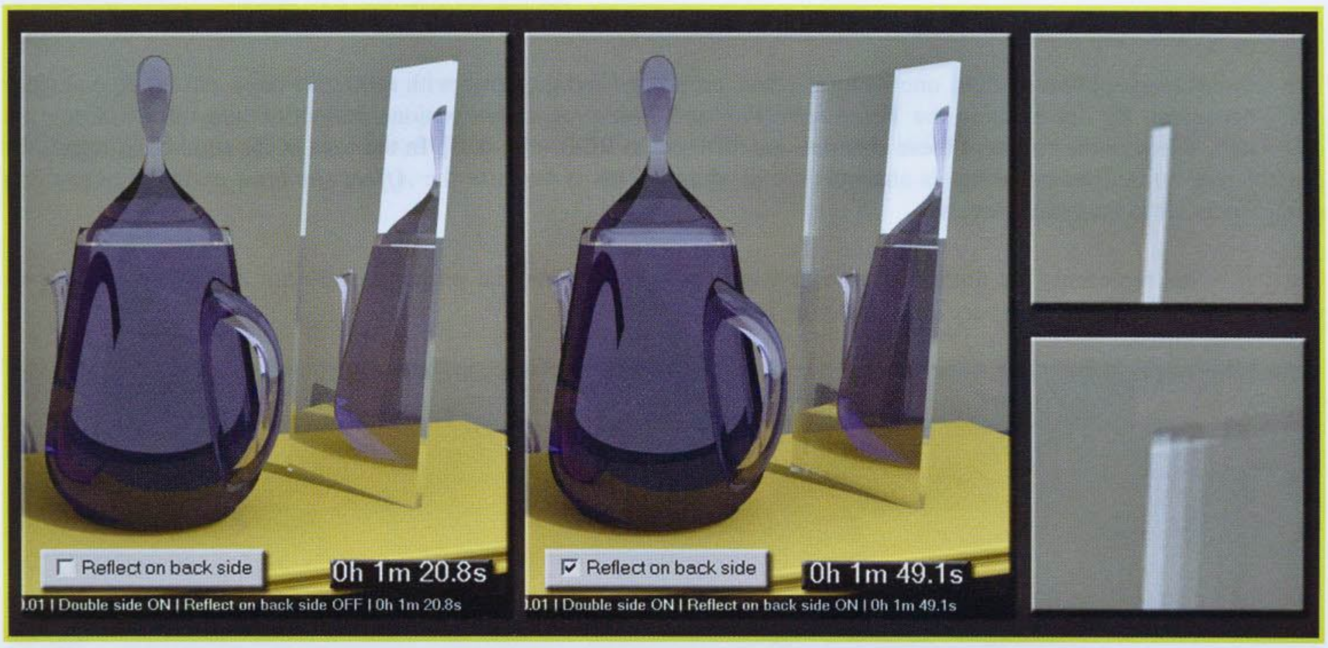
Double-side - if this is active, *VRay* also renders the back faces of single polygons without thickness which are usually not calculated.



■ **Figure 5.112**
A plane without thickness, in order to be properly rendered on both sides, must have a material with the ***Double-side*** option activated.

The geometry of the test is a plane with no thickness, coiled over on itself and made with a reflective material. If the ***Double-side*** option is not activated, the face which is opposite to the normal will not be correctly rendered. In the left-hand example, the reflection on the back of the geometry is not taken into account. Only ***Diffuse*** color is calculated.

Reflect on back side - when active, reflections are calculated also for back faces of geometries.



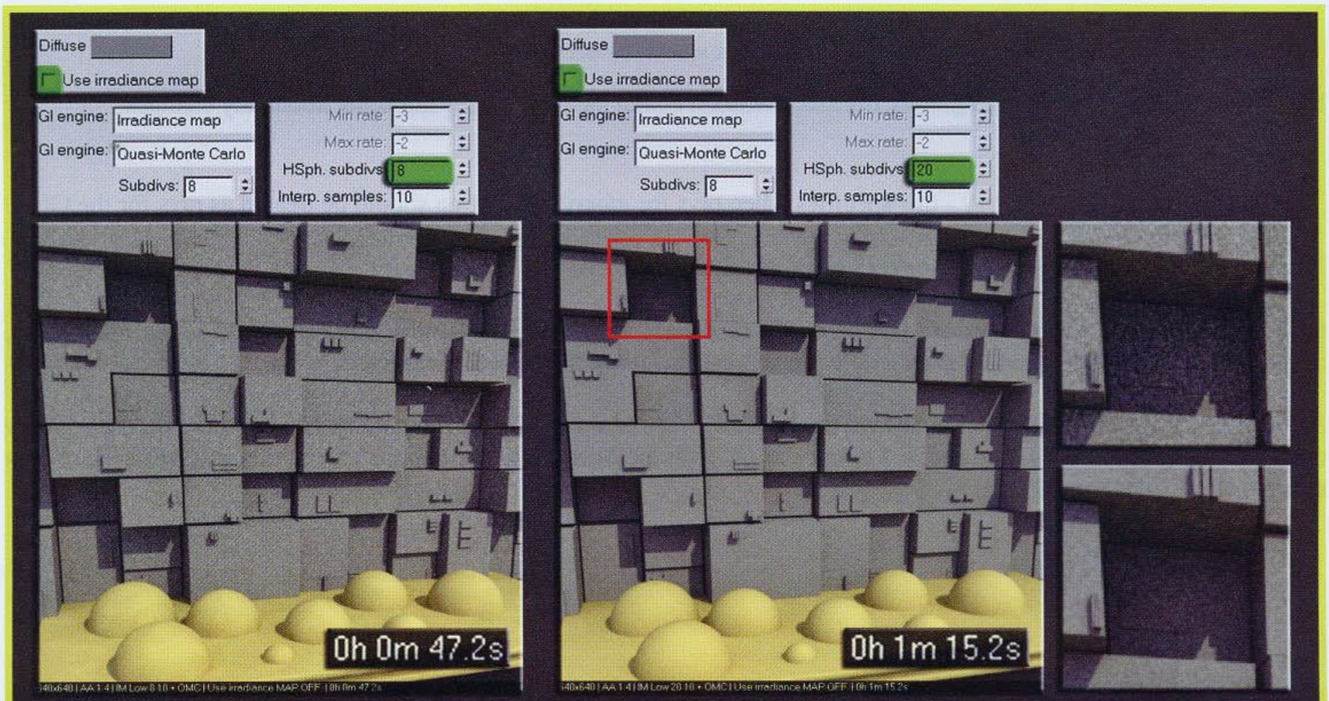
■ **Figure 5.113**
Improvement of reflections on transparent materials thanks to ***Reflect on back side*** being activated.

By activating this option, faces which are internal or at the back the normal of a geometrical object have reflective properties. This factor increases realism, but also rendering times because the number of reflections increases.

Use irradiance map - when active, this parameter makes *VRay* use ***Irradiance Map*** for the approximation of indirect illumination of the material. Otherwise the ***QMC*** method is used. Its deactivation is useful if one is using the ***IM*** for ***Primary bounces*** and the geometry, rich of details, is not correctly rendered via the biased method.
In order to understand this parameter one must observe the render in the following scene, calculated both with the ***IM+QMC*** method and the ***QMC+QMC*** method. In both tests the ***Use irradiance map*** parameter is of no use at all.

■ **Figure 5.114**
IM+QMC and ***QMC+QMC***. The ***Use irradiance map*** parameter is always active.

Nothing new in this image. The left-hand one is more uniform but has more artifacts thanks to the low quality of the ***IM***, whereas the right-hand image is more detailed but has more noise. Now we will move on to render using only the ***IM+QMC*** method, using different ***Hsph Subdivs*** values and deactivating the ***Use Irradiance map*** parameter.



■ **Figure 5.115**
IM+QMC. The parameter ***Use irradiance map*** is not active.

In this second example, compared to the previous one where the ***IM+QMC*** method was used, the only difference is the deactivation of the ***Use irradiance map*** parameter for the grey material used for the wall. Moreover at first ***Hsph. Subdivs*** is set to 8 and after to 20. One can notice how for the wall GI calculation is no longer left to the ***IM*** but to the ***QMC***, which has a quality level based on the ***Hsph. Subdivs*** setting found in the ***IM***. The orange pavement and its orange sphere geometries are also managed by the ***Irradiance map***. So to sum up, it is possible, with the ***IM+QMC*** GI calculation method, to make sure that some geometrical objects are exclusively handled as if the calculation method was ***QMC+QMC***. ***QMC Subdivisions***, which are only globally defined, are exclusively handled by ***IM*** subdivisions. This can be useful during animations, where for very detailed geometries one can decide the use of ***QMC*** with the aim of avoiding flickering problems.

For example what would happen if we used the ***QMC+LC*** method?

■ **Figure 5.116**
QMC+LC. The ***Use Irradiance map*** parameter is deactivated but does not influence the render at all.

In this case, as the ***IM*** is not being used, the ***Use irradiance map*** parameter does not bring about any changes.

Treat glossy ray as GI ray - this specifies when rays for ***glossy*** calculation should be treated as rays for the GI:

Never - ***glossy*** rays are never treated as GI rays;

Only for GI rays - ***glossy*** rays are treated as GI rays only after the GI has been calculated. By using this option, rendering is faster when there are ***glossy*** reflections present. This is the default parameter.

Always - ***glossy*** rays are always treated as GI rays. As a result, the ***Secondary GI engine*** is used for ***glossy*** calculation. If, for example, one decided to use the ***LC*** with the ***Secondary engine***, the calculation of ***glossy*** effects can take advantage of this method.

Energy preservation mode - this parameter allows one to determine the mutual influence between ***Diffuse*** color and reflection or refraction color. In its calculations, *VRay* maintains a certain balance between an object's incident energy and reflected energy, exactly like in real life. This, for instance, is the reason why reflection influences and balances the amount of diffused and refracted light. A 100% reflection completely removes both ***Diffuse*** color and refraction. Vice versa, a 100% refraction completely removes color from an object. This parameter establishes in what way this should be applied and can do so separately for each RGB component or according to the level of monochromatic intensity.

RGB - this mode calculates attenuation on RGB channels. Thus a material which is completely white with a red reflection produces a cyan color surface. This is the default method.

Monochrome - this mode calculates the attenuation according to the intensity of ***Diffuse***, ***Reflection*** and ***Refraction***.

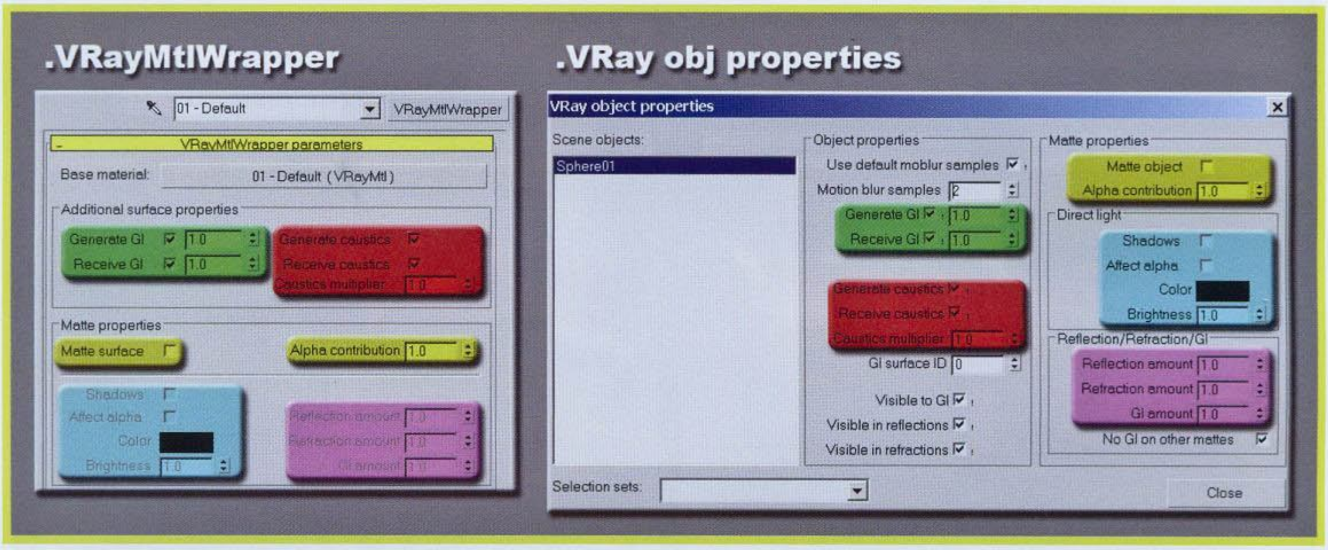
■ Figure 5.117
The difference between ***Energy mode: RGB*** and ***Energy mode: Monochrome***.

By using the monochrome system, *VRay* renders the reflection of the same color found in the ***Reflect*** channel.

VRay Material : VRayMtlWrapper

INTRODUCTION

This shader consists of a rather special material, in the sense that it could be described as a "utility" material. Thanks to its use, one can add special properties to a basic material. These new parameters should be familiar, as they re-propose all of what the ***VRay object properties*** panel offers.



■ **Figure 5.118**
Comparison between the ***VRayMtlWrapper*** material and the ***VRay object properties*** window.

As one can notice, the parameters are identical, the only difference being that the ***VRayMtlWrapper*** applies these values through the association of a material, whereas the ***VRay object properties*** does so via single geometries. Notice how material settings overwrite those used in the ***VRay object properties*** window. So, for example, if caustics generation was deactivated for the sphere object via its properties, by assigning a ***VRayMtlWrapper*** and activating the caustics feature, caustics would be generated by the sphere in any case.

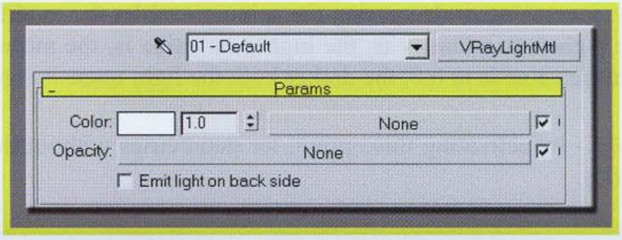
The way this material works is very simple. The first parameter, the ***Base material***, can be any material: a ***VRayMtl***, a Standard material, a ***VRayLightMtl***, and so on. The ***Base material*** represents the actual material which gives a geometrical object its typical traits, such as color, specularity and refraction. When this material is placed within a ***VRayMtlWrapper***, interesting properties are immediately added to it. If the ***VRayMtlWrapper*** parameters were left to default values, the Base material behaves exactly as if the ***VRayMtlWrapper*** had not been added. Vice versa, it is possible to modify its features, which will then influence all the geometries associated with the material.

As all the parameters of this shader have been handled during the description of the properties of the ***VRay object properties*** window, we shall not go over them a second time. One can refer to the chapter concerning the ***VRay: System rollout***.

VRay Material : VRayLightMtl

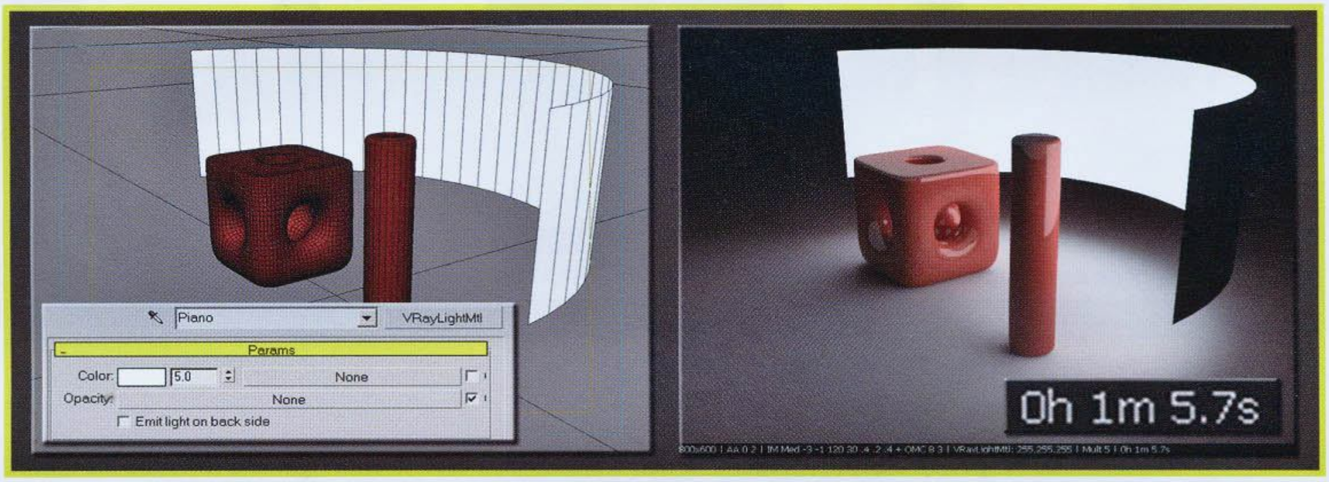
PARAMETERS

Like all the main rendering programs, *VRay* can generate illumination via light sources, such as 3ds Max, Spot Light, Direct Light, Omi Light and ***VRayLights***, light sources which are to be covered in the next chapters. It is also possible to get *VRay* to use any geometry as a light source, having assigned a particular shader to it, namely ***VRayLightMtl***, a material which allows any mesh to emit light.



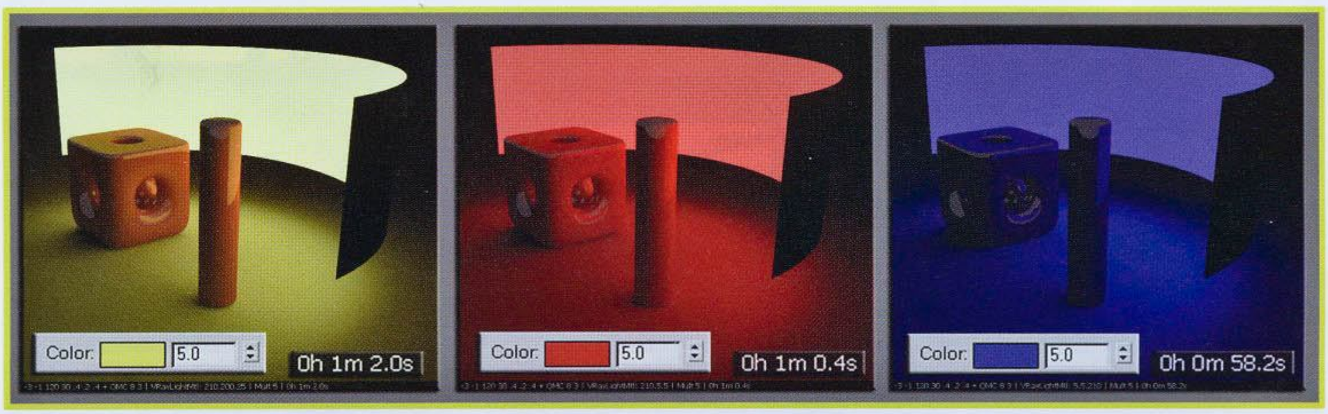
■ **Figure 5.119**
The ***VRayLightMtl*** material.

With this particular shader one can assign a color or map to the object, which emits light by transforming the mesh into a light source.



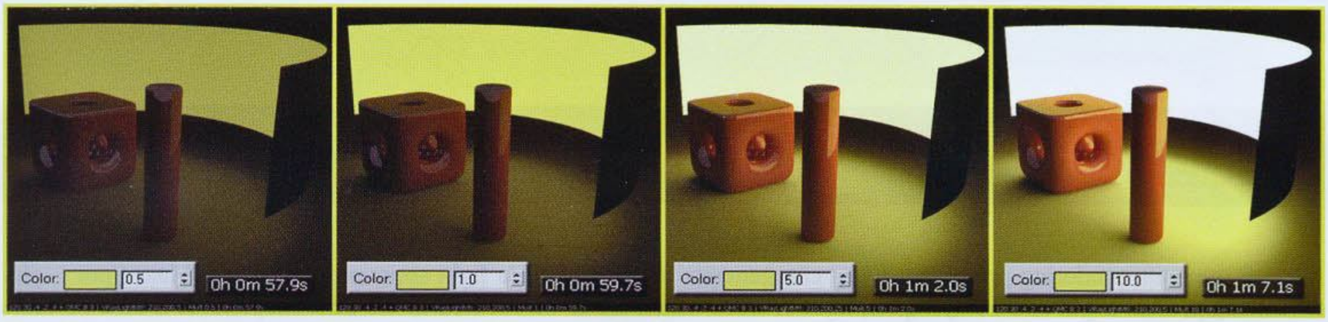
■ **Figure 5.120**
Example of use of the ***VRayLightMtl*** material. Quality is very good and time demand is low.

Color - this allows one to decide what color to associate to material and therefore to define the type of light produced.



■ **Figure 5.121**
Use of different colors for the emitter geometry.

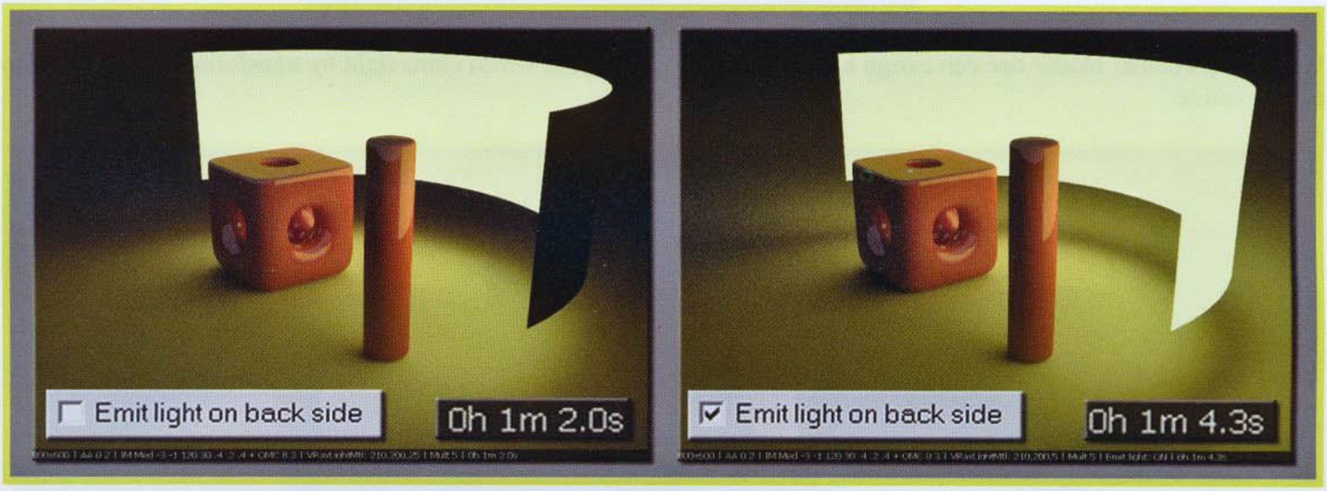
Multiplier - this parameter multiplies the amount of color intensity used and therefore controls the amount of light emitted.



■ **Figure 5.122**
Variation in the intensity of emitted light.

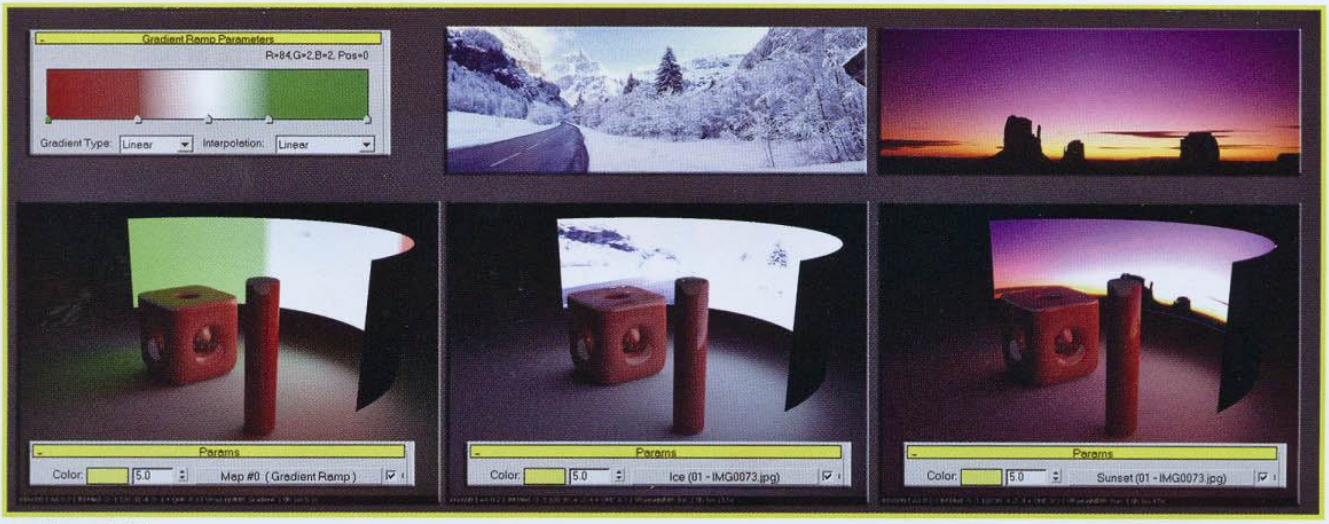
As well as varying intensity, increasing or decreasing the spinner enables the modification of the color itself of the self-illuminating geometry. The higher its intensity is, the higher its value is, the more the color of the plane tends toward white. Vice versa, the closer the value is to 0.0, the darker both the light intensity and the color of the plane become.

Emit light on back side - by activating this option, for geometries without thickness (as in this case), the emitter generates light from both sides.



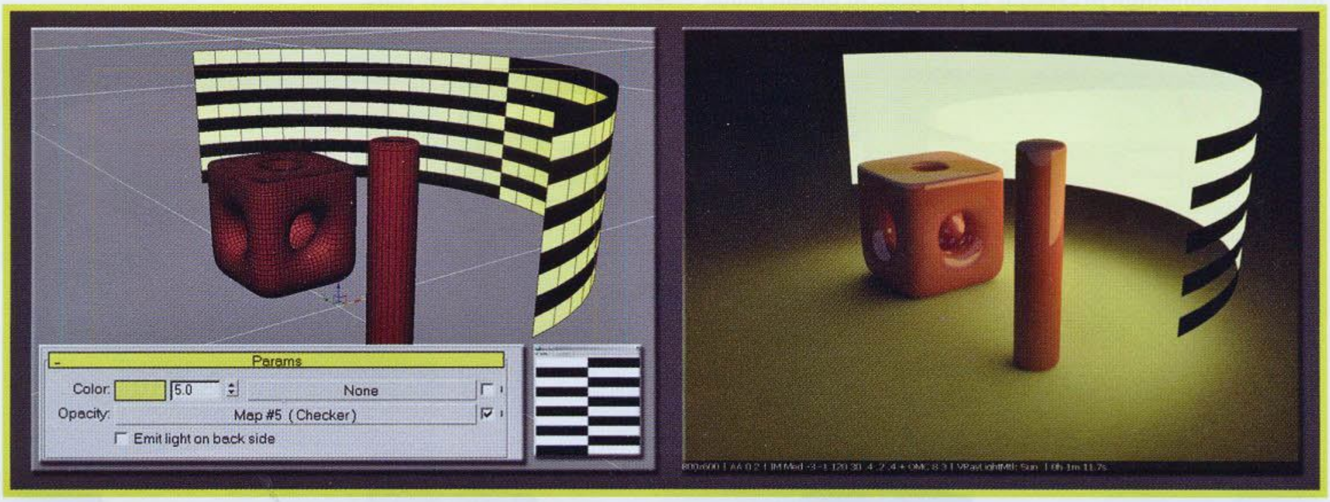
■ **Figure 5.123**
Emission of light on both sides of the plane.

TextMap - it allows one to specify a map, procedural or not, which is then used to emit light and texturize the object. If one wishes to modify intensity, it is enough to alter the ***Multiplier*** spinner.



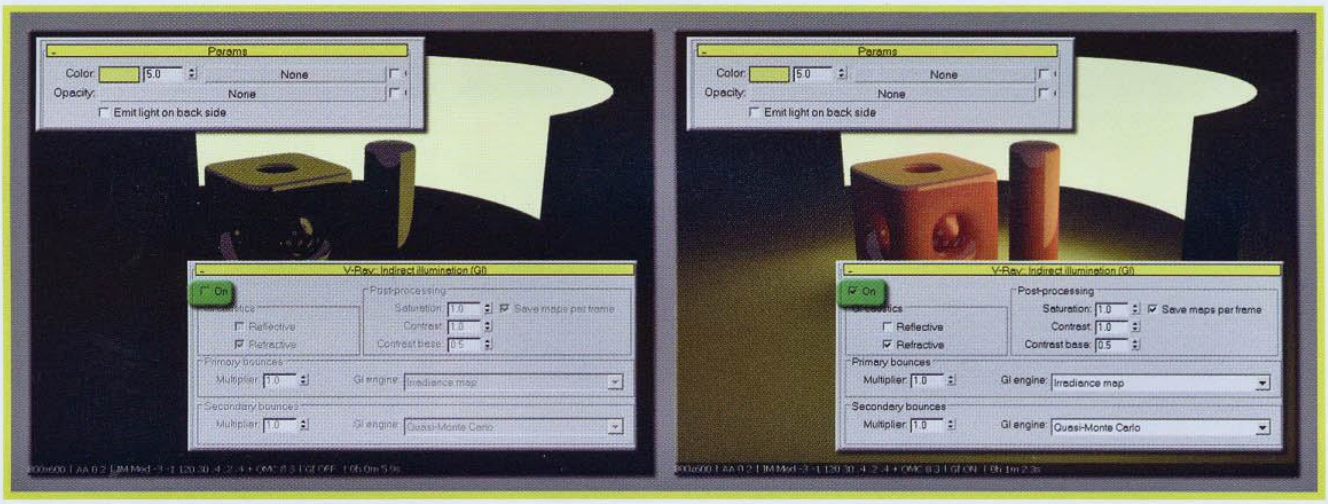
■ **Figure 5.124**
Textures being used for light generation.

Opacity - this establishes in what manner the back side of emitters is handled. The black parts of textures make objects completely transparent. On the contrary, white parts make it opaque.



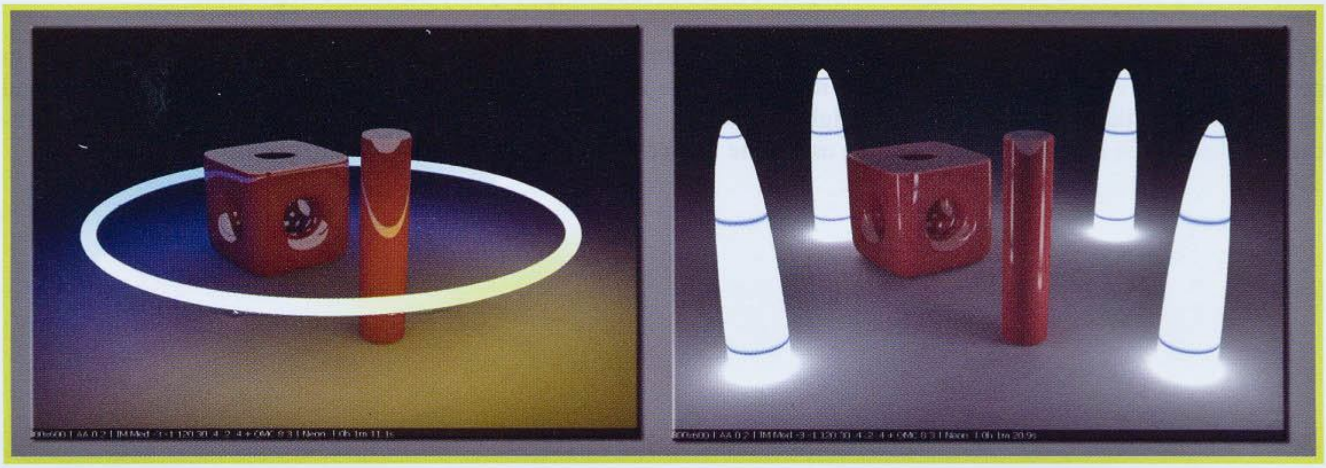
■ **Figure 5.125**
By placing the same map in both slots, opacity would be apparent in the front face also.

In order to enable a geometrical object to emit light, it is necessary to activate Global Illumination. Without doing so the plane would merely be a fluorescent geometry without luminous capacity.



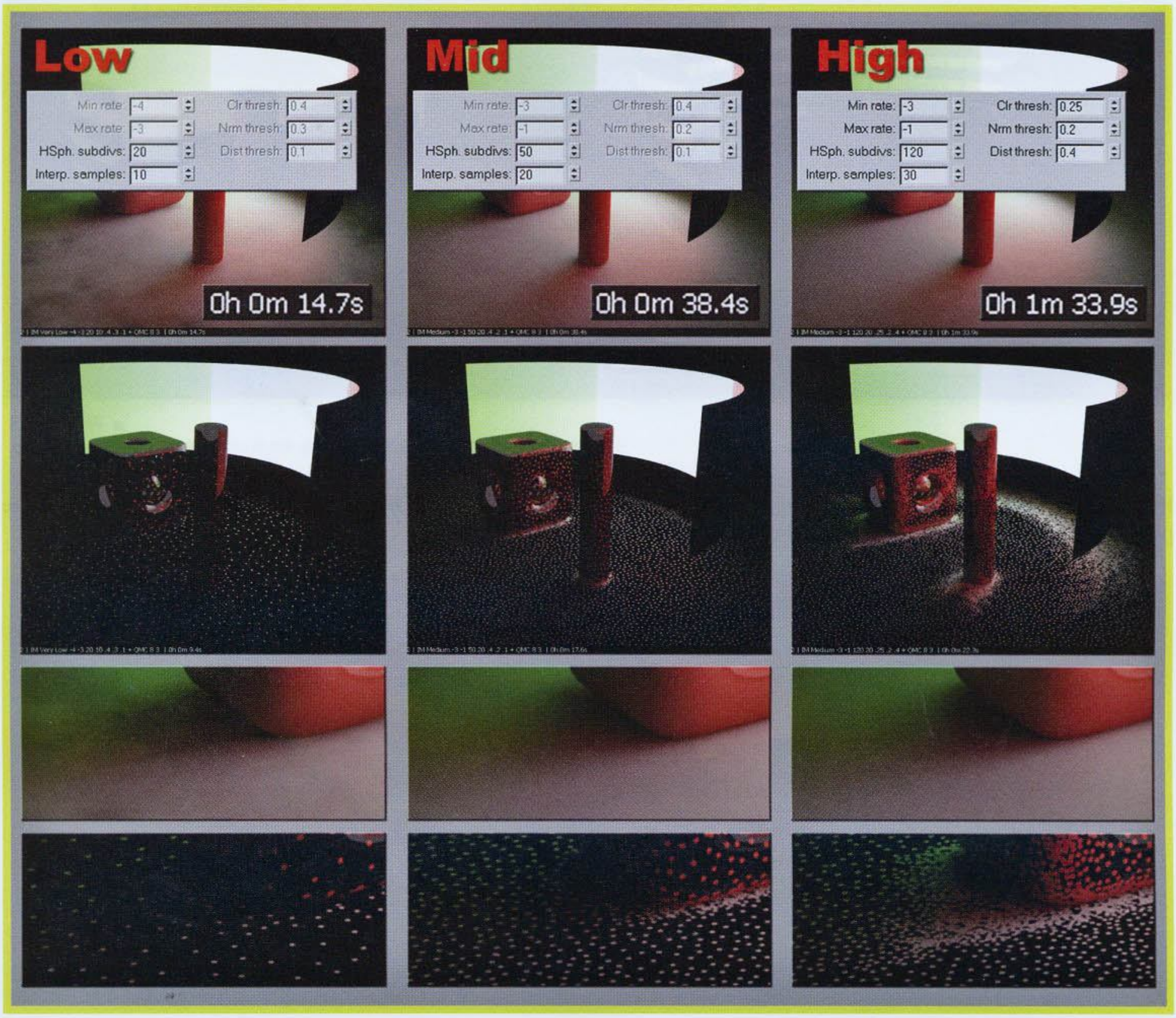
■ **Figure 5.126**
In order to obtain a luminous effect of an object with a ***VRayLightMtl*** applied to it, one must activate the GI.

As well as simple planes, be they straight or curved, one can use any other geometry, like for example tubes for simulating neon lights or other geometrical objects.



■ **Figure 5.127**
Some examples of self-illuminating geometries.

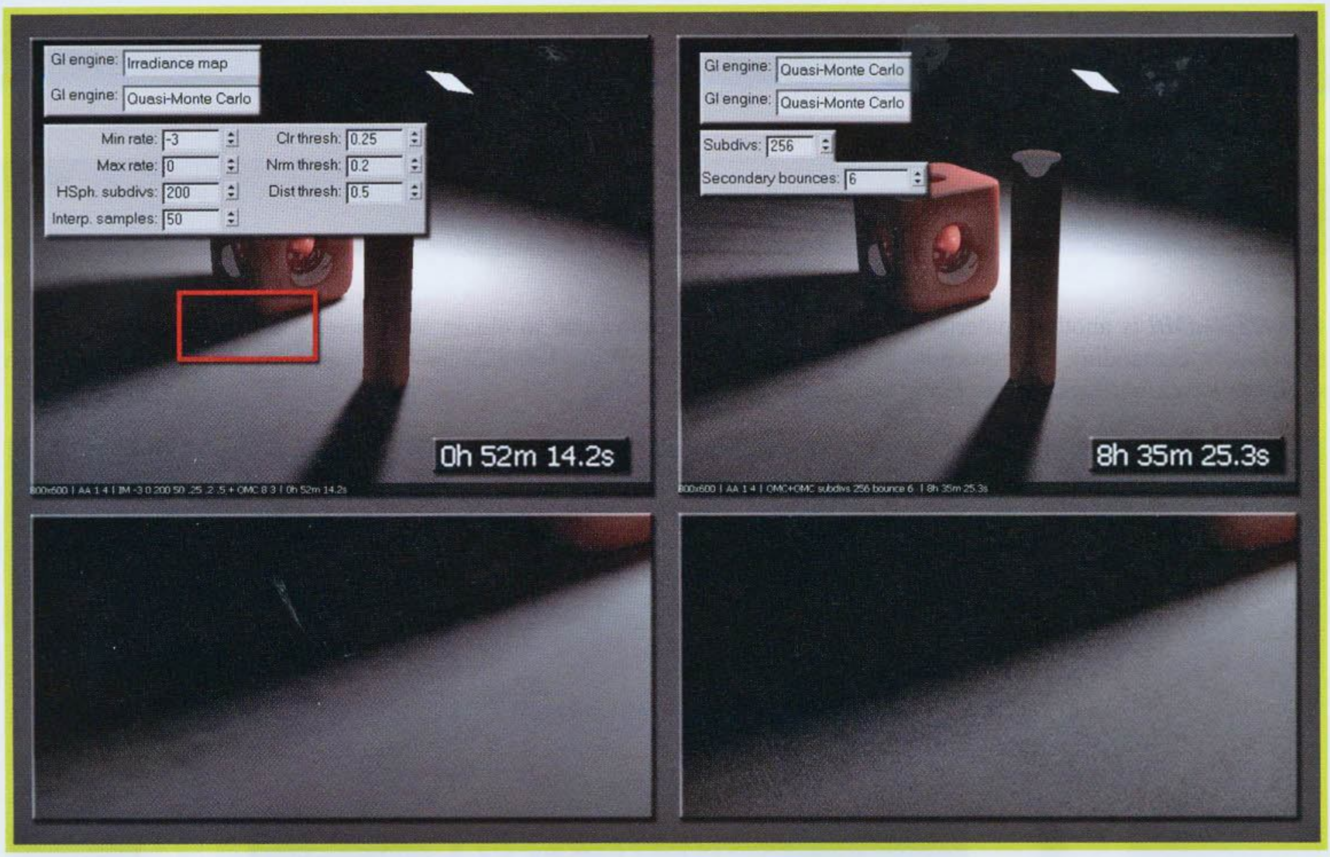
The main problem of self-illuminating materials lies in the management of shadow quality. As one may have easily noticed so far, there is no parameter which allows the regulation, for instance, of ***Subdivisions***. Illumination quality and therefore shadow quality is instead handled by the GI. In the previous cases the ***IM+QMC*** method has been used. By using medium-high ***presets***, it is possible to obtain much cleaner renders in a short time.



■ **Figure 5.128**
The quality of shadows generated by self-illuminating materials.

It is clear how the quality of shadows generated by emitters is closely connected to the quality of the GI. The more samples *VRay* has available for GI calculation, the better the shadow quality and the render in general.

There are however some limitations in the use of self-illuminated materials, such as for instance the impossibility to generate photons with a ***Photon Map***. Also it is difficult to attain satisfactory results with very small self-illuminated objects. If this is the case, even with very high GI settings, shadows can have many problems. In order to obtain well-defined shadows, it is advisable to avoid the use of the ***VRayLightMtl*** assigned to very small geometries.



■ **Figure 5.129**
A scarcely efficient use of ***VRayLightMtl*** applied to a small geometrical object.
- - - DVD - - -

By spending one hour and using very high settings for such a simple scene, with the ***Irradiance map*** one obtains low quality shadows. Even spending 9 hours with the ***QMC+QMC*** method, the most precise method there is, results are not satisfactory. Shadows are too dirty and noise too high, even with 256 ***Subdivisions***!

One can thus conclude that self-illuminated materials are excellent in situations where one wants to obtain illumination with very smooth shadows with medium/large-size objects, or when it is necessary to fulfill objects such as neons.

VRay Material : VRay2SidedMtl

INTRODUCTION

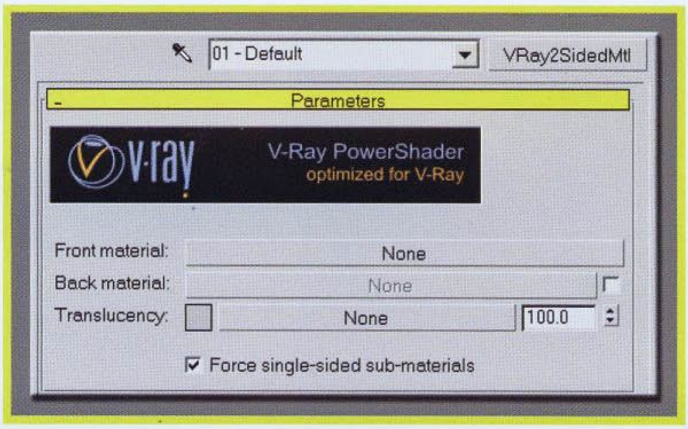
VRay2SidedMtl is another material available in *VRay*. It allows one to obtain translucency effects in a short time, a notoriously demanding phenomenon in terms of resources. With *VRay2SidedMtl*, thanks to its features, one can simulate leaves, curtains, lamps, fabric and sheets of paper.

Figure 5.130
Some photographs of elements which can be simulated via ***VRay2SidedMtl***.

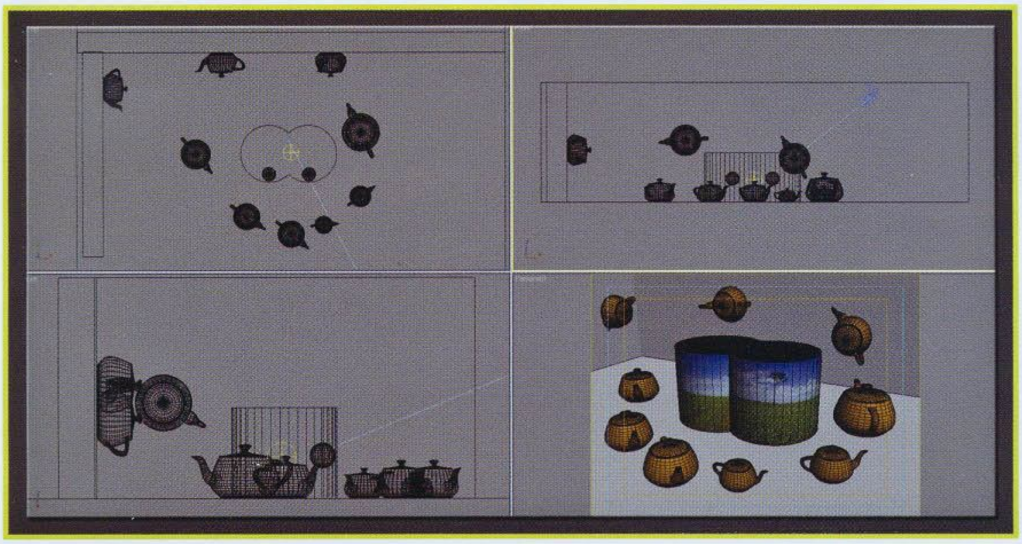
PARAMETERS

VRay2SidedMtl is a simple shader and very few parameters are available for it in *VRay*.

Figure 5.131
The ***VRay2SidedMtl*** shader and its parameters.

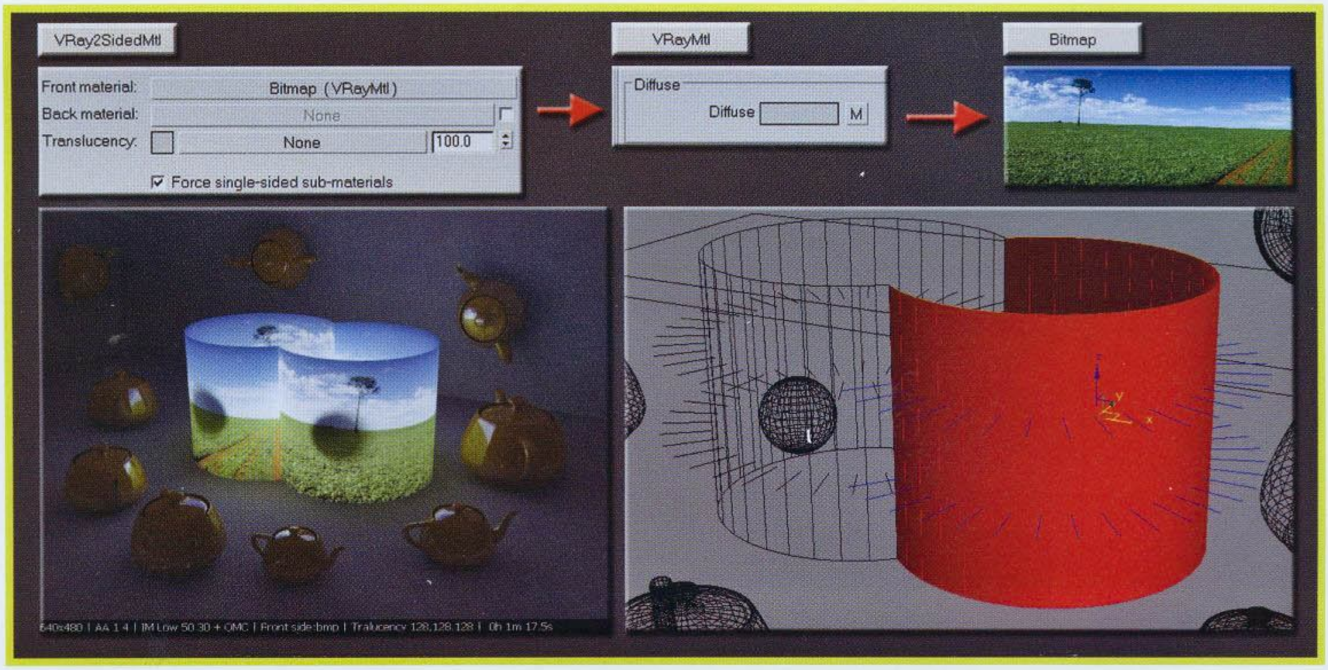


The scene used for the next few examples consists of a plane with two vertical walls as its boundary, on which a series of teapots have been placed which are illuminated by a ***VRayLight***. The light source has been placed within a curved geometry which has no thickness. A ***VRay2SidedMtl*** has been assigned to these curved planes. Two spheres have been placed inside. This will allow us to analyze the translucency phenomenon better.



■ **Figure 5.132**
The scene used for the ***VRay2SidedMtl*** test.

Front material - the material inserted into this slot will be used for texturizing the front faces of the geometry. The orientation of polygons can be recognized by observing their normals.



■ **Figure 5.133**
The ***VRay2SidedMtl*** being used, with ***Translucency*** =[128,128,128]. One can see the silhouettes of the shadows of the internal spheres projected from within.

The material used as a ***Front material*** is a simple ***VRayMtl*** with the ***Diffuse*** channel taken up by a Bitmap. The material is applied to outer faces, but as no ***Back material*** is active, both the inner and outer face have the same material. A very important role is played by the ***Translucency*** setting, as we will see shortly.

Back material - in this slot one can insert a material which is then used for texturizing the back faces of the geometry. Both in the previous and following example, a mild ***skylight*** is used as well as the illumination given off by the ***VRayLight*** within the geometry. This helps to make both sides of the mesh visible, even when translucency is missing.

■ **Figure 5.134**
Back material being used. In the right-hand example ***Translucency*** color has been changed to black.

By activating ***Back material*** one can assign two different materials to the same face. In the left-hand test, translucency is set to RGB = [128, 128, 128], the default value. On the right instead, it has been switched off in order to observe the effect of ***Back material*** better. Also, ***skylight*** enables one to observe the front, which is texturized with the Bitmap. If it has not been enabled, the outer part will appear completely dark, allowing one to see the Checker procedural map alone.

■ **Figure 5.135**
Skylight turned off with ***Translucency*** settings = [0,0,0] and ***Back material*** activated.

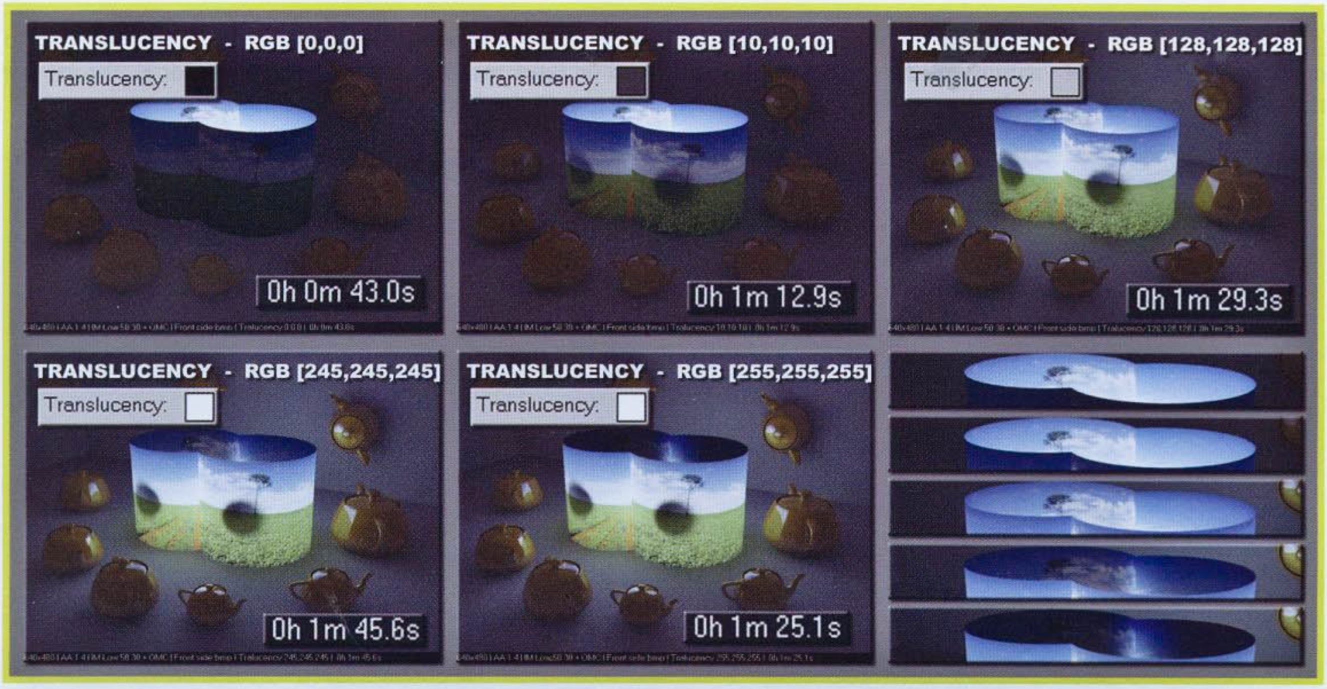
Check box - if this has been deactivated, *VRay* considers both faces of the geometry as if they were texturized via the ***Front material***. If it is activated, one can use the ***Back material***.

Translucency color - with the color selector one can determine the amount of translucency. It is also possible to determine the degree of visibility of the two materials, the ***Front*** or ***Back*** materials. The darker the color is, the greater the visibility of the ***Front color*** becomes. If it is completely white, only the ***Back material*** is visible.

■ **Figure 5.136**
Variation of the ***Translucency*** parameter applied to a ***VRay2SidedMtl*** with both ***Front*** and ***Back material*** active. Rendering time is very low and the general quality is very good, without any problems or strange blotches.

With RGB = [0, 0, 0] there is no translucency. By increasing it even just a little, one starts to notice the first signs of ***SSS*** and both of the materials of the two faces are visible. With RGB = [5, 5, 5] the ***Front material*** already starts to disappear. With RGB = [32, 32, 32] it has practically disappeared and translucency is the only thing visible, due to the ***Back material***. With RGB = [128, 128, 128] we have the perfect balance between the outer and inner part, whereas with ***Translucency*** RGB = [255, 255, 255] we obtain maximum illumination and maximum translucency.

Most of the time one will not find one needs to use two different maps for the front and back side of a geometrical object. Leaves, fabric, curtains and so on usually have the same material on both sides.



■ **Figure 5.137**
Translucency effect with only the ***Front material*** active.

Translucency slot - by inserting a map in this slot one can decide which areas of the geometry show the ***Front Material*** or ***Back material***. As we have already seen for other effects, such as the ***Reflection*** and ***Refraction*** channels and others, with a simple map it is possible to specify the position and intensity of the ***SSS***, whereas with its spinner one can alter its effect on the material. With a setting value of 100, the ***VRay2SidedMtl*** only uses the map of the ***Translucency*** channel for the ***SSS***, whereas with a setting value of 0.0 it will use the color selector. Intermediate values allow both to be used.

Force single-side sub-material - In order to render ***SSS*** effects correctly when using the ***VRay2SidedMtl***, it is necessary for sub-materials used for defining the ***Front*** or ***Back material*** to have the ***Double-side*** option inactive. This parameter is located in the ***rollout: Option*** of the ***VRayMtl***. If all the sub-materials have this option activated, thanks to ***Force single-side sub-material***, *VRay* will overwrite settings by automatically deactivating ***Double-side*** for all the materials. It is active by default

■ **Figure 5.138**
For a correct use of the ***VRay2SidedMtl*** one must deactivate the ***Double-side*** option of the ***Front*** or ***Back Mtl***. Otherwise one can activate the ***Force single-side*** parameter.

In the left-hand image, even with a very high level of translucency, the geometry does not allow light to pass through. This is because of two factors. Firstly the ***VRayMtl*** used as a Front material has the ***Double-side*** option active. Secondly, the ***Force single-side sub-materials*** is deactivated. It is enough for one of the two parameters to be set differently for the ***SSS*** to become visible, allowing light to pass through.

It is important to make a last observation on the geometries. In order to exploit the features of the ***VRay2SidedMtl*** to the full it is best to use a mesh which is completely without thickness. It is in any case possible to use thick objects, although in some cases this can lead to very long rendering times. In the test being carried out, the use of geometries with thickness does not bring about any variation in calculation time.

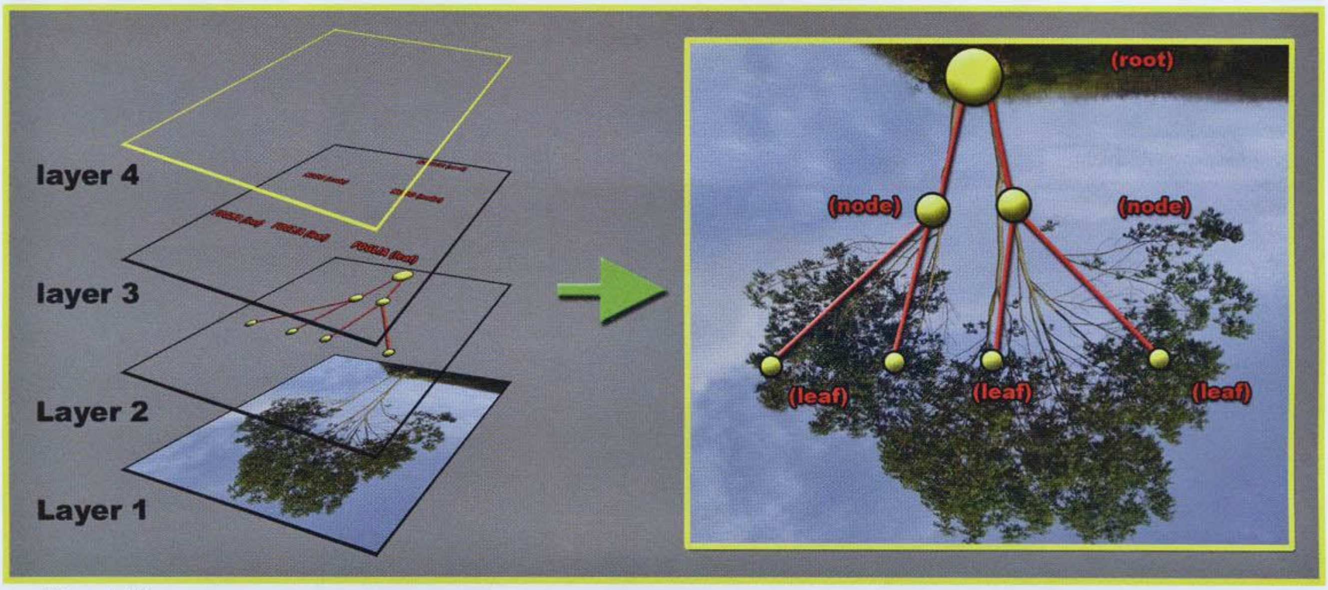
■ **Figure 5.139**

VRay2SidedMtl being used on solid geometries with variable thickness.

VRay Material : VRayBlendMtl

INTRODUCTION

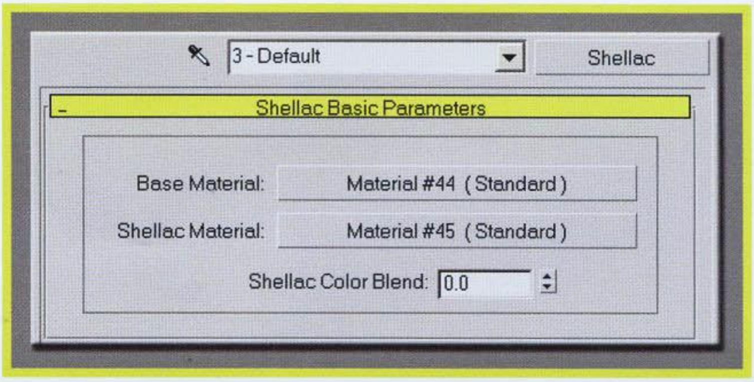
VRayBlendMtl can be used to fulfill materials with "layers". For example these shaders can be made up of more than one *VRayMtl* laid over each other. The percentage of use of each one can be manually defined by the user. Anyone using programs such as AutoCAD or Photoshop has definitely heard the term "layer" before. Layers can be compared to the transparent sheets used in traditional technical schemes. They can be placed over each other, with the function of organizing and managing information of various projects.



■ **Figure 5.140**
Graphic elaboration made with layers.

In the previous examples the four layers, each one transparent except for the first one, permit the image to be created by placing them over each other. In 3ds Max one can generate shaders, materials with the same principle, by using the *Shellac* material.

■ **Figure 5.141**
The *Shellac* material, a 3ds Max shader.

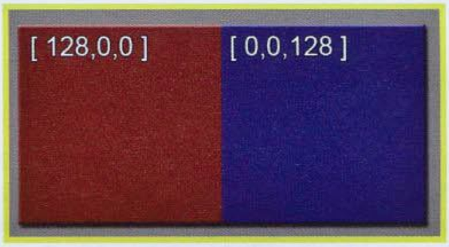


With this material one can unite or superimpose two or more materials, just like one does with Photoshop layers. The *Base material* can be defined as layer 1, whereas the *Shellac material* represents layer 2. By placing other *Shellac* shaders within the *Shellac material*, one can use infinite layers. For the next exercise only two main materials will be used: *Base material* and *Shellac Material*.

■ **Figure 5.142**
Shellac material being applied with a ***VRayMtl*** with a Checker procedural map applied to it used as Base material, and for the ***Shellac Material*** another ***VRayMtl*** with a uniform color RGB=[255,255,0] applied.

If we take a look at the image we notice the change of color in the maps. But how does 3ds Max superimpose these two maps? By summing the sub-material ***Shellac material*** with the ***Base material***.

If one analyzes the Checker texture, the colors used are tones from known RGB coordinates.

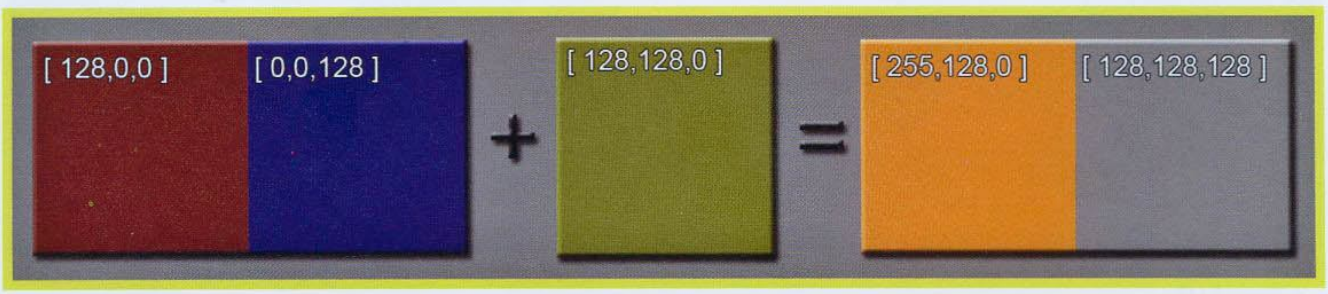


■ **Figure 5.143**
Two colors of the ***Checker*** map.

The yellow color used for the ***Shellac material*** instead, is equivalent to RGB = [128, 128, 0].

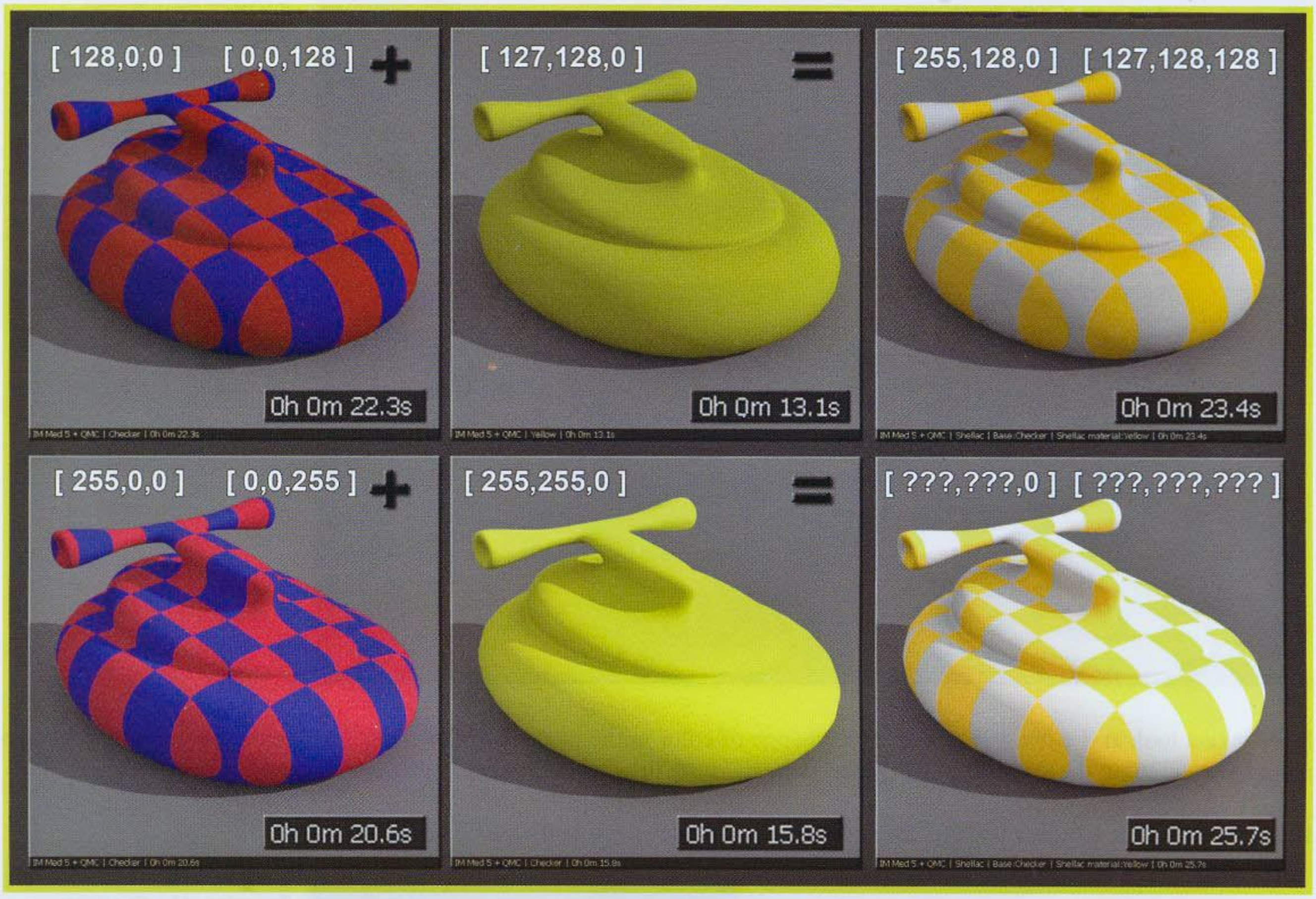
■ **Figure 5.144**
The yellow color used as ***Shellac Material***.

By mathematically summing together the RGB colors of the Checker map with the colors of the ***Shellac*** map, something very interesting happens:



■ **Figure 5.145**
Algebra sum of the RGB values carried out by the ***Shellac*** shader.

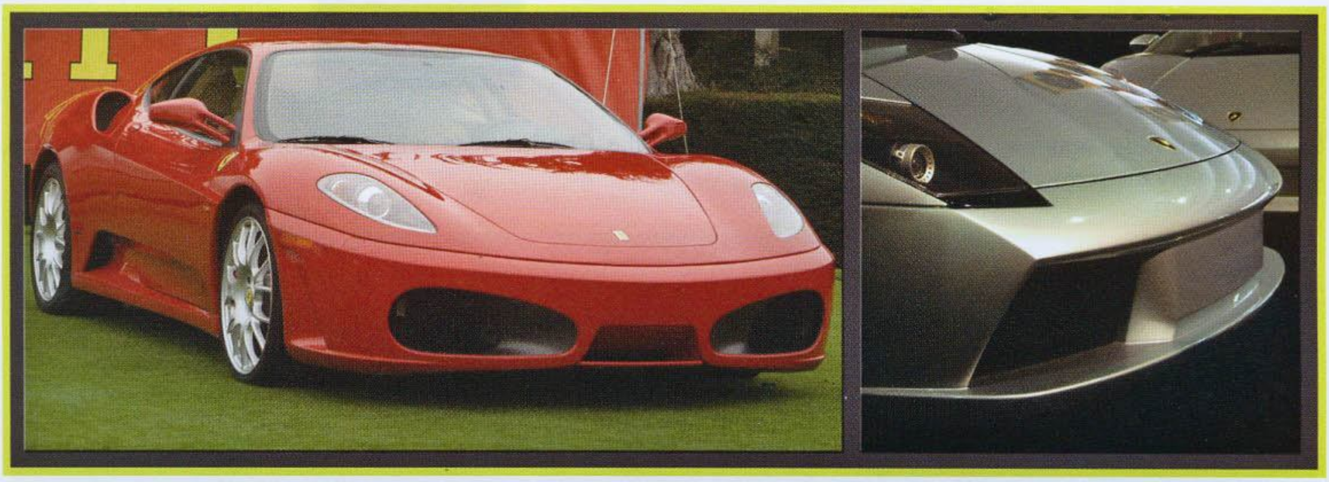
Shellac material is very predictable. In order to obtain the final color it is enough to sum the base and ***Shellac*** colors. But a problem arises. What happens if one uses RGB colors with a sum result which exceeds RGB = [255, 255, 255]?



■ **Figure 5.146**
The ***Shellac*** material and problems with the mathematical representation of colors.

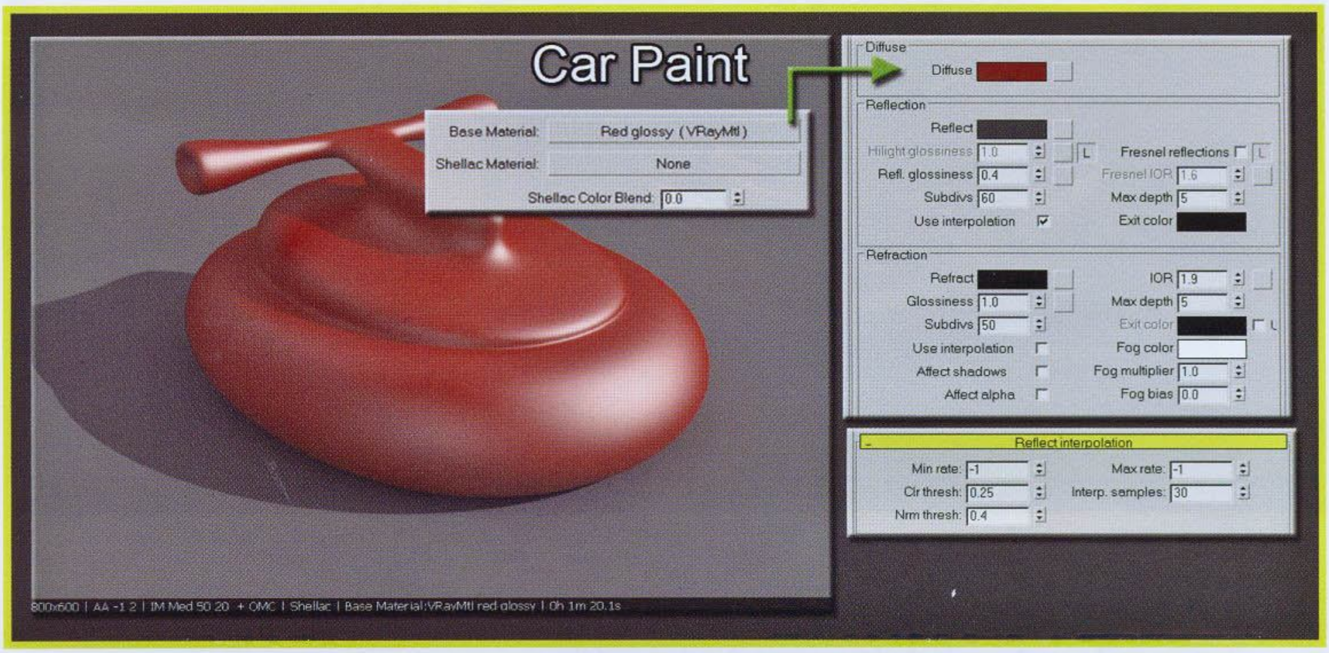
By using colors which give a sum greater than RGB = [255, 255, 255], one might expect 3ds Max to "clip off" values exceeding RGB = [255, 255, 255]. In other words the shader would not take the "extra color" into consideration. In this case the geometry should therefore be completely white. Instead this is not what happens. The result we get resembles a sort of self-illumination. There is no actual light produced in the true meaning of the word, but the sensation one gets when looking at it is similar. Soon we will see how to solve this problem.

The ***Shellac*** material, produced by the famous ***Blur Studios*** and later included in ***Kinetix 3DStudio MAX 2***, has always been used for precise objectives, and the most famous of these is for creating the effect of metal-effect paint for vehicles, commonly called Car Paint.



■ **Figure 5.147**
Car Paint: its simulation in 3ds Max is ensured by the presence of the ***Shellac*** shader.

Technically, for creating a realistic and complex material like car paint, ***Shellac*** can be a valid helper. First of all, one starts by using a ***VRayMtl*** for the ***Base material*** with a lot of ***glossy*** effect. This gives us the blurred reflections and ***Highlights*** that Car Paint requires.

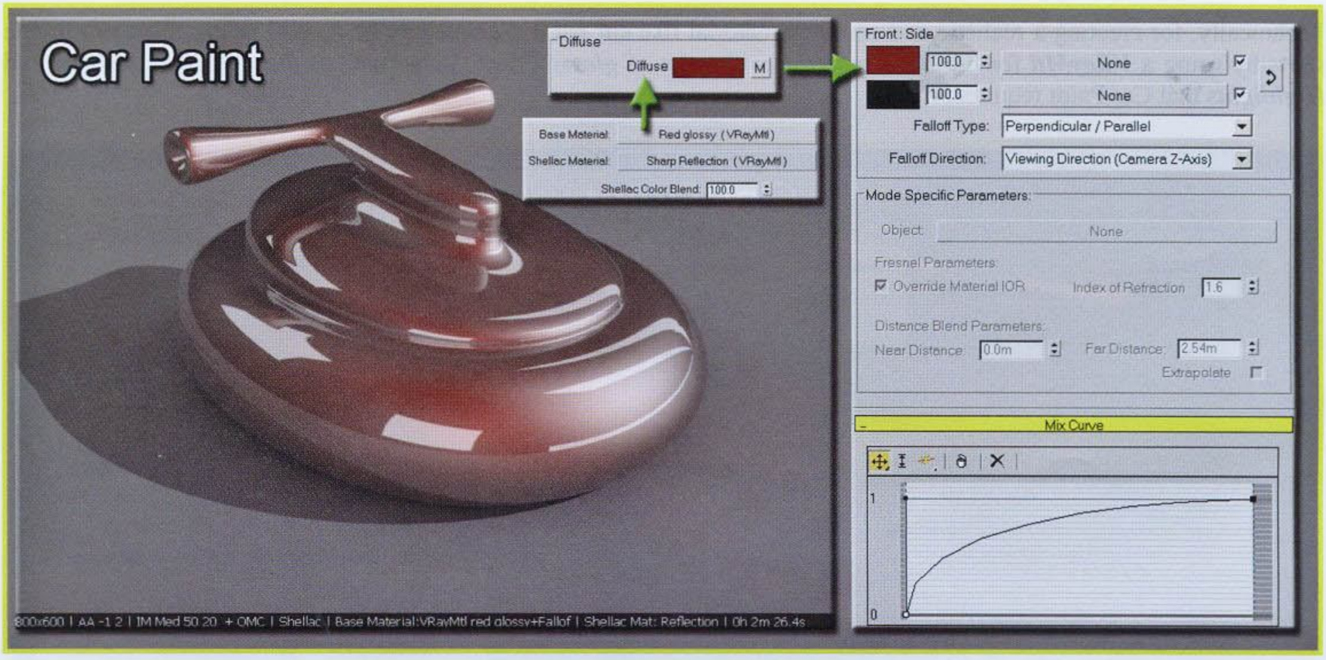


■ **Figure 5.148**
The first "layer" of Car Paint material.

Soon the second layer, ***Shellac material***, is added, which gives the material a shiny quality.

■ **Figure 5.149**
Adding on the second layer: reflection.

By analyzing the second layer, one realizes that this material only adds one type of information: Fresnel reflections. No color, just a second reflection channel. Thanks to the ***Shellac*** material it has therefore been possible to assign two different types of reflections to a single shader, one with ***glossiness*** and one without. In order to improve Car Paint further, one can assign to the diffuse channel of the ***Base Material*** not a solid color but a Falloff map. This will help give the paint that metalized look, whereby colors tend to blend if observed parallel to the point of view.



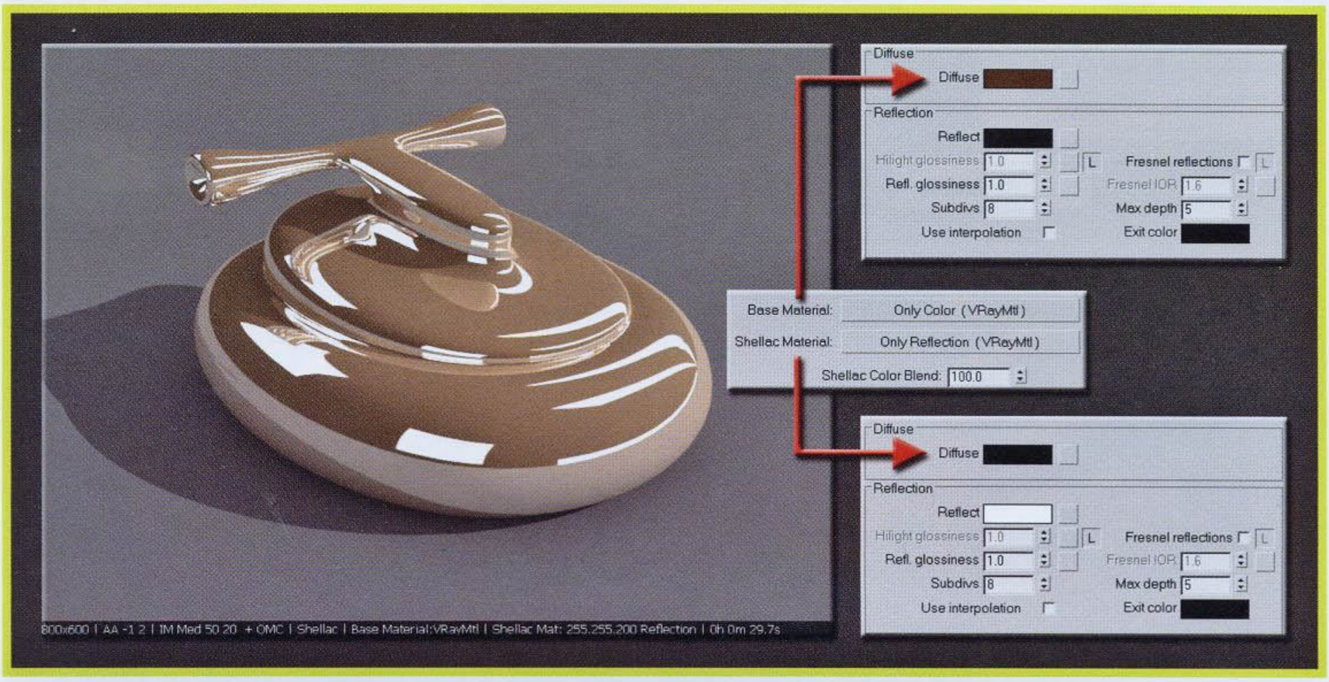
■ **Figure 5.150**
Car paint shader complete.

Obviously the more layers there are, the longer rendering time will be.

■ **Figure 5.151**
The three phases of Car Paint creation. Notice how rendering time increases even adding on a single layer.

The other parameter of the ***Shellac*** material, ***Shellac Color Blend***, defines the percentage of use of the sub-material ***Shellac Material***. Its range goes from 0 to infinite. In most cases, one tends to not go beyond 100%. The reason concerns the observations we made earlier on the sum of colors. For instance if we sum [128, 0, 0] + [127, 128, 0] with ***Shellac Color Blend*** = 100, the final result of the shader is RGB = [255, 128, 128]. If ***Shellac Color Blend*** were equivalent to 50, the result would be similar to RGB = [191,64,0]. If ***Shellac Color Blend*** was equal to 200, the result would be RGB = [382, 256, 0], an RGB color which is impossible to replicate.

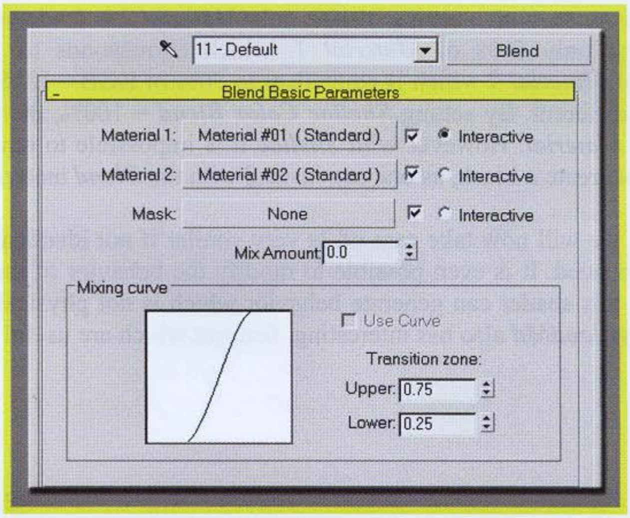
There is also another feature which somehow also concerns the adding phenomenon of the ***Shellac*** material and it is described in the next example.



■ **Figure 5.152**
The classic problem due to using the ***Shellac***.

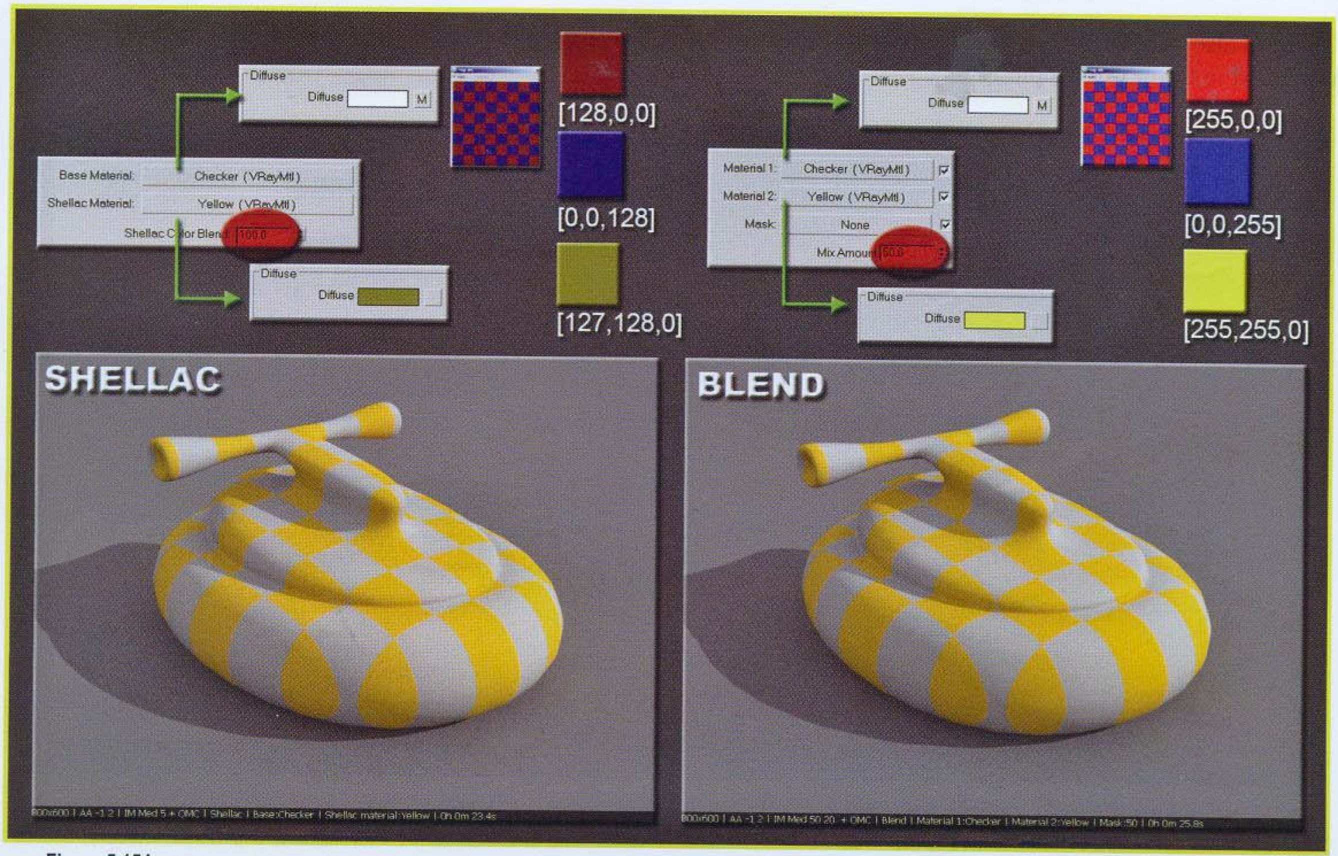
In the ***Base Material*** a ***VRayMtl*** material has been inserted with only one modified diffuse channel. The ***Shellac Material*** is also a ***VRayMtl*** with maximum reflections. At this point one should already have understood that the render, as it is suggested, cannot be correct. A material with 100% reflection cannot have any color, as mirrors or chrome materials do not. From the physical point of view, this material cannot exist in nature!

Another shader in 3ds Max which allows one to blend two or more materials is the ***Blend*** material.



■ **Figure 5.153**
The interface of the ***Blend*** material.

This material is very similar to ***Shellac***. The difference lies in the possibility of mixing the two maps, ***Material 1*** and ***Material 2***, not only via a percentage like ***Mix Amount*** for the ***Shellac Color Blend***, but also via a map called ***Mask***. We have stated that ***Blend*** material is similar to ***Shellac*** and in actual fact, with an accurate calibration of settings, one can obtain the same results.



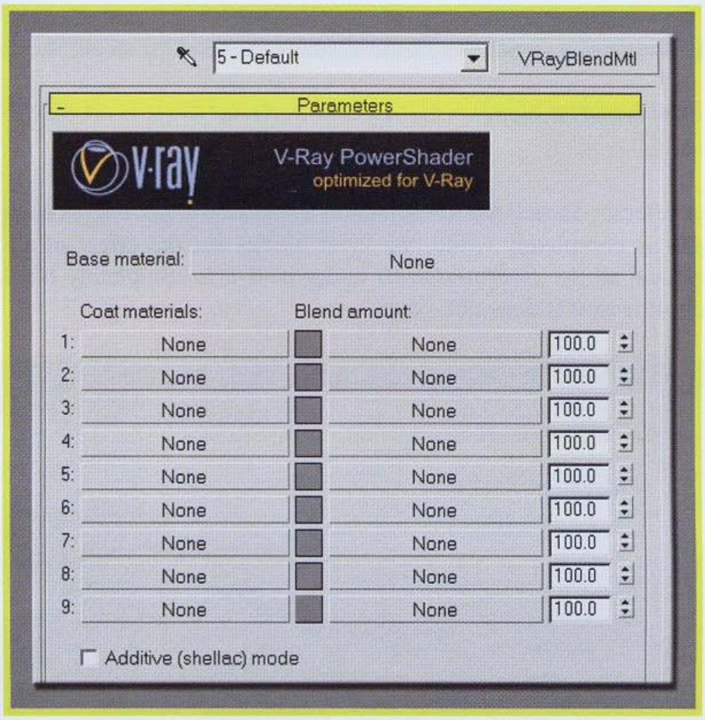
■ **Figure 5.154**
Comparison between ***Shellac*** and ***Blend*** material.

As far as the ***Shellac*** material is concerned, the subject has been amply covered in the previous pages. It is now necessary to focus our attention on the ***Blend*** material and above all understand how it works. The first observation that can be made is that also this material, in order to blend maps, works with color addition, just like ***Shellac***. But how is it possible for the same result to be obtained by using different colors? The answer lies within the ***Mix Amount*** parameter. This parameter allows one to regulate the amount of ***Material 1*** and ***Material 2*** used. If ***Mix Amount*** is set to 0, only ***Material 1*** is rendered. Vice versa, with a setting of 100, only ***Material 2*** is rendered. In the case we are examining it is set to 50. This means that only 50% of ***Material 1***, which corresponds to 50% of RGB = [255, 0, 0] and RGB=[0,0,255], and 50% of ***Material 2*** which is equivalent to 50% of RGB = [255, 255, 0] is added up. With ***Shellac*** the procedure was slightly different. By setting ***Shellac Color Blend*** = 100%, the whole ***Shellac Material*** was added and not mixed to the ***Base Material***. However with ***Shellac*** it is impossible to render only the ***Shellac Material*** sub-material and also one cannot create a Mask, as one can instead with the ***Blend*** material.

The ***VRayBlendMtl***, which we will now take care of, is very similar if not identical to the Blend material in 3ds Max, which we have briefly introduced. It is even possible to modify the behavior of the ***VRayBlendMtl*** by replicating the ***Shellac Material***, although this shader can generate behavior which is not physically correct. Lastly it is important to emphasize the fact the ***VRayBlendMtl*** also has interesting features which are useful in post-production processes.

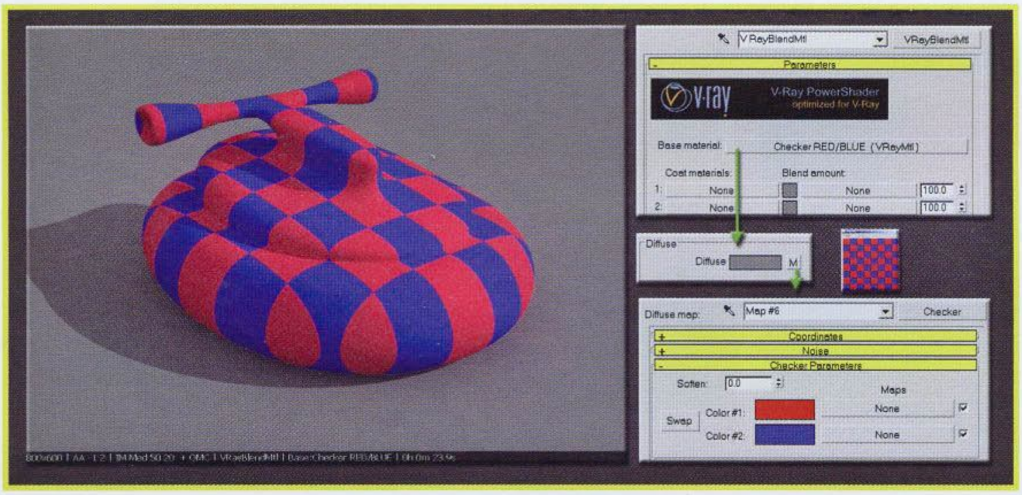
PARAMETRI

The ***VRayBlendMtl*** can be used to make materials made up of layers, thus obtaining very complex shaders. It has been conceived to work only with the ***VRayMtl***, ***VRayOverrideMtl***, ***VRay2SidedMtl***, ***VRayFastSSS***, ***VRayLight*** or other ***VRayBlendMtl*** materials. In calculations it is faster than ***Shellac***, ***Blend*** and ***Composite*** materials, it maintains physical accuracy and moreover by using the ***Render Elements: VRayMtlSelect***, one can separate the various sub-materials of the ***VRayBlendMtl*** in single channels.
Similarly to the ***Blend*** material in 3ds Max, ***VRayBlendMtl*** uses a base material to which other layers are added after, by superimposing their effects in a scalar way. A first layer, ***Coat material #1***, is added to the ***Base material***, which can for instance be a ***VRayMtl*** with ***Diffuse*** represented by a solid color. Imagine another ***VRayMtl*** with maximum reflection. After another layer is added to the material resulting from the union of the ***Base material*** and the ***Coat material #1***, namely ***Coat material #2***, which has a texture in the diffuse channel.



■ **Figure 5.155**
The interface of the ***VRayBlendMtl*** material. This material, which is not compatible with the Standard materials in 3ds Max, has in any case been projected in order to host any material compiled with the API (Application Program Interface) of VRay.

Base material - it is the base material to which various layers can be added. If one does not specify otherwise, the material used by default is made of a completely transparent material.



■ **Figure 5.156**
Base material being used on its own.

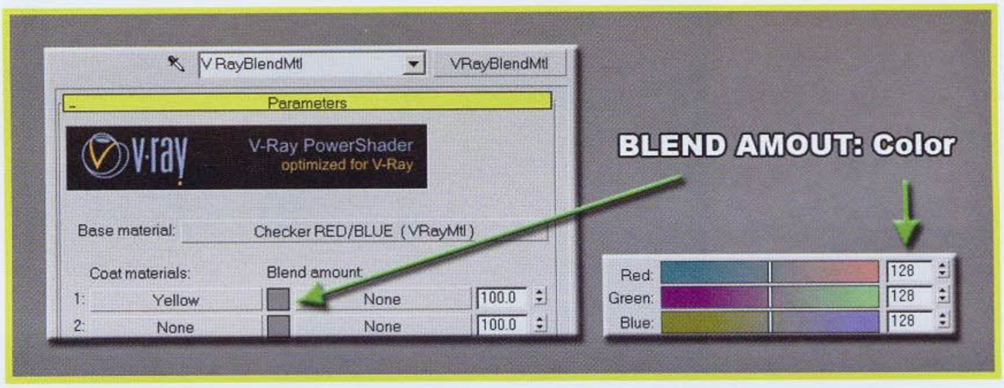
In this test we have simply used a ***VRayBlendMtl*** with a ***VRayMtl*** as a ***Base material***. The ***VRayMtl*** has a Checker map in the ***Diffuse*** channel. In this sense, ***VRayBlendMtl*** is nothing more than a simple material represented by a ***Base material***. The color red corresponds to RGB = [255, 0, 0] and blue instead to RGB = [0, 0, 255].

Coat material - this represents the various layers of the ***VRayBlendMtl***.

■ **Figure 5.157**
A first level being added to ***Base Material***.

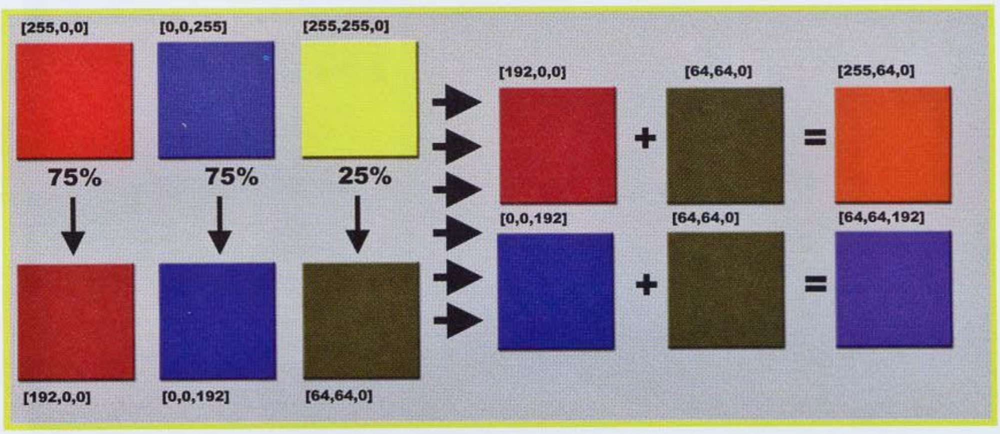
By adding a first layer, the ***Base material*** blends with the ***Coat material #1***. It is a ***VRayMtl*** with ***Diffuse*** corresponding to RGB = [255, 255 ,0].

Blend amount - the color selector associated to each ***Coat material*** allows one to decide the influence of each layer on the final result. If the color is completely white, the layer blocks out the effect of all layers above it and the final result includes the layer itself and all the ***Coat materials*** under it. If it is completely black, the ***Coat material*** does not influence the final result at all. It is as if it does not exist.



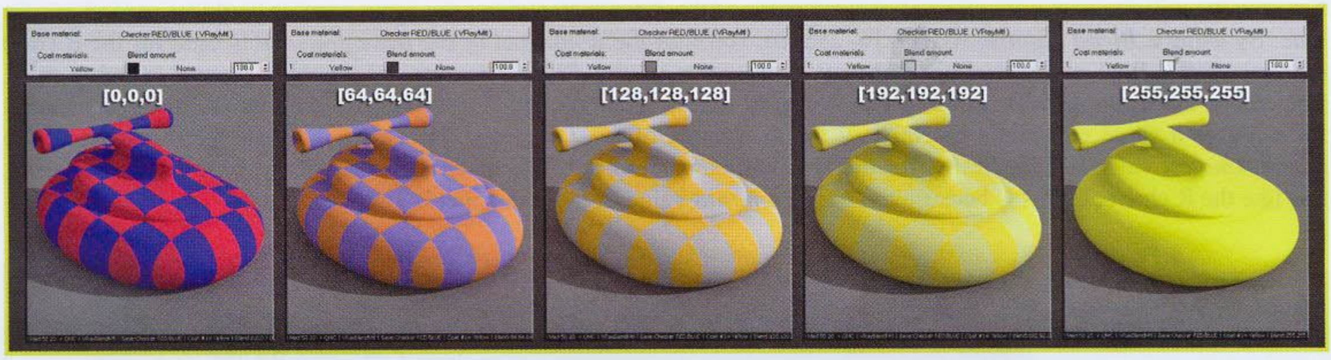
■ **Figure 5.158**
The color selector allows one to define the amount of blending between the ***Base material*** and the various layers called ***Coat materials***.

Similarly to what happens for the ***Blend*** and ***Shellac*** material of 3ds Max, ***VRay*** sums the RGB components of materials. As the percentage of blending is 50% by default, that is, RGB = [128, 128, 128], ***VRay*** uses half the RGB value for the ***Base material*** and half the RGB value for the ***Coat material*** in order to obtain the final result.



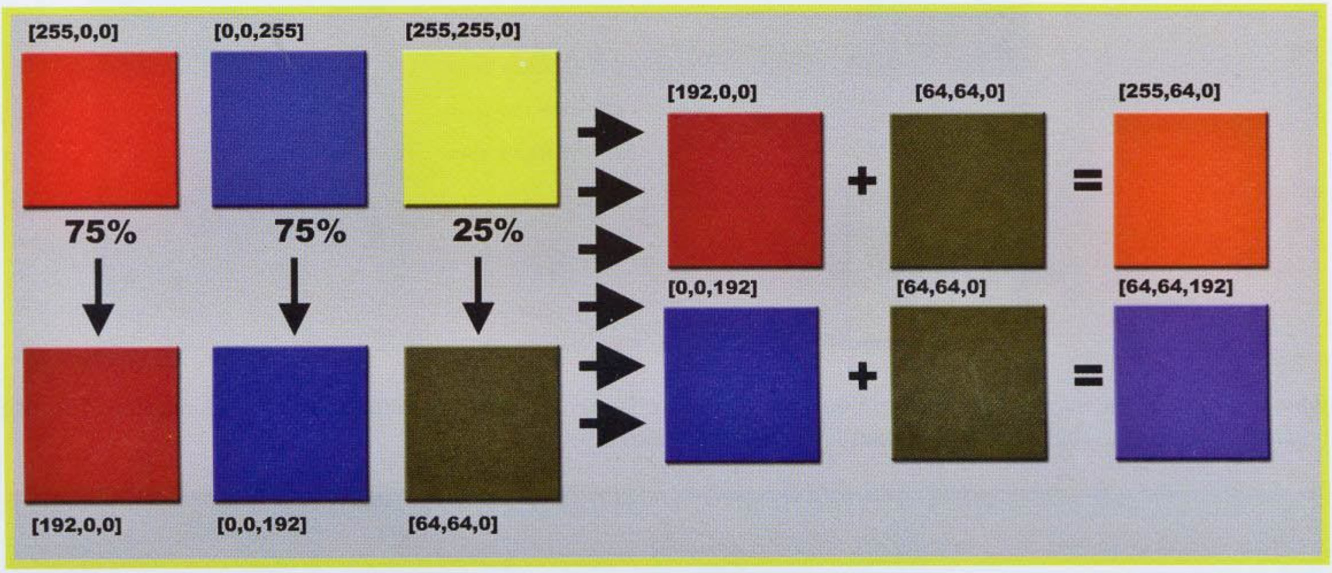
■ **Figure 5.159**
Algebra sum of RGB components, with a blending color of RGB=[64,64,64].

By using tones of grey it is therefore possible to determine the influence of the ***Base material*** and single layers on the final result.



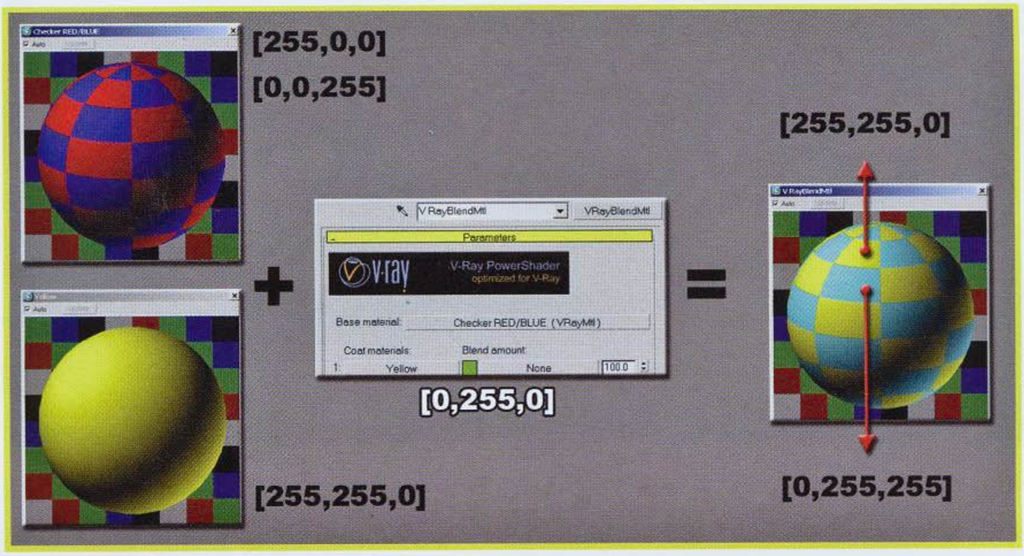
■ **Figure 5.160**
Blend Amount set to different tones of grey.

If one wanted to use ***Blend Color*** RGB = [64, 64, 64], the final result is made up by ¾ of the ***Base material*** and by ¼ of the ***Coat material #1***.



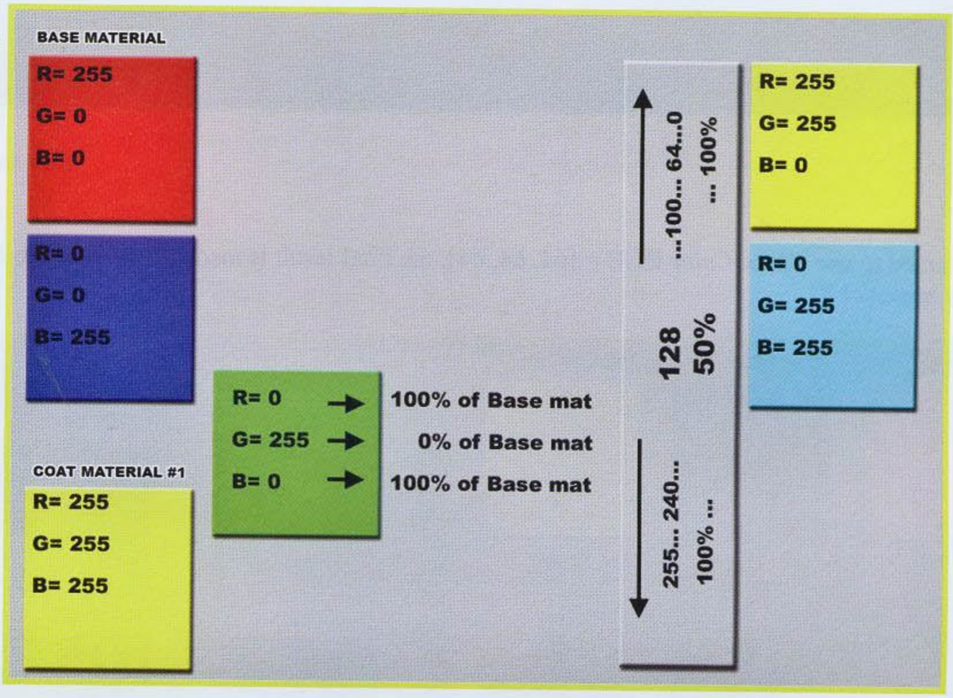
■ **Figure 5.161**
Active blending with RGB = [64, 64, 64]. The render is brighter because of the illumination in the scene.

But what happens if one uses a color other than grey for the ***Blend amount*** parameter? If this is the case it will be necessary to separate each R,G,B channel and take the respective quantities in function of the ***Blend amount***. We will give an example by using ***Blend Amount*** RGB = [0, 255, 0], a bright green. In this case, ***VRay*** returns a checker pattern with a color of RGB=[255, 255, 0],[0, 255, 255].



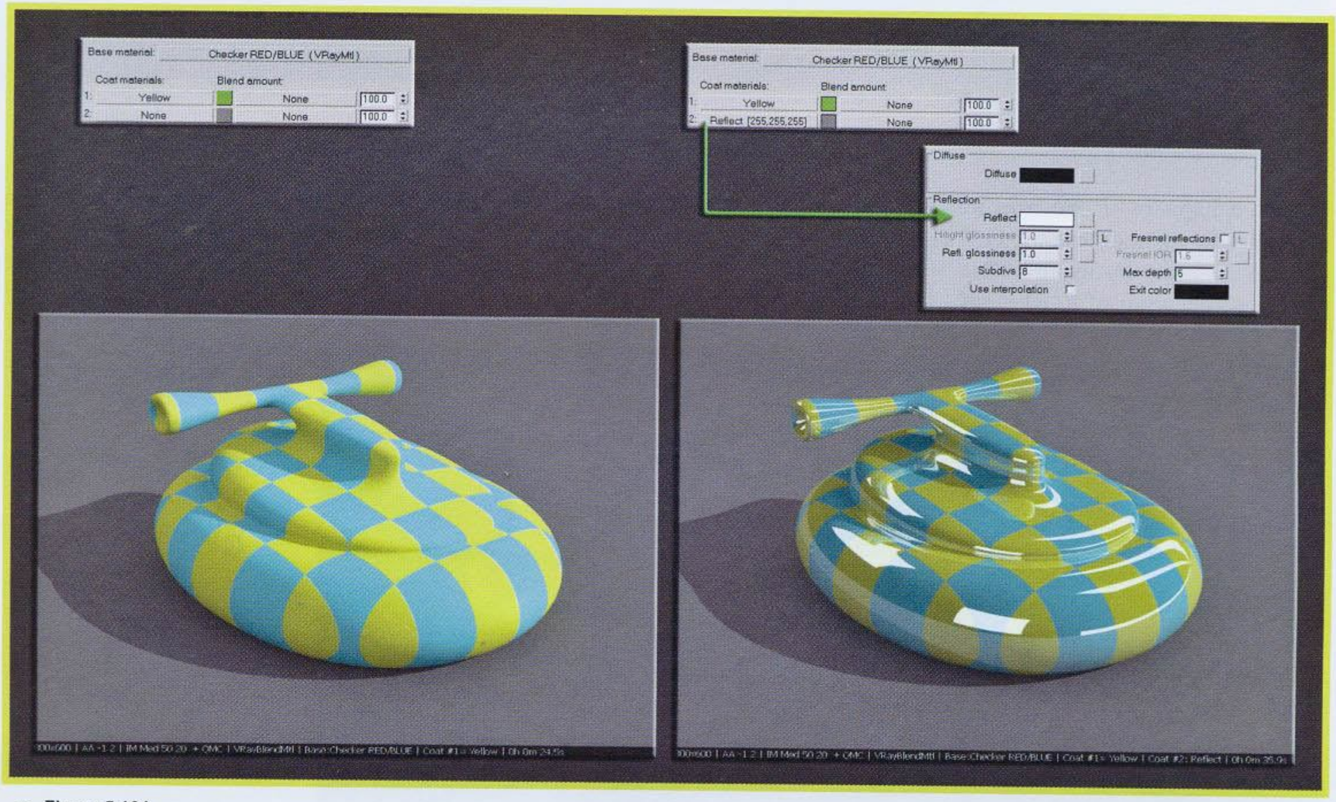
■ **Figure 5.162**
Blend Amount represented by a non-neutral RGB color.

For now we will focus our attention on the red color of the ***Base material*** but the same process could be carried out for blue. By using a ***Blend amount*** RGB = [0, 255, 0], 100% of the R channel of the ***Base material*** is used for the R (red) channel of the final material. This is because the color of the ***Blend amount*** has for this channel a value of 0. (remember the example which uses a scale of grey tones). The color of the G (Green) channel of the final material instead uses a value equivalent to 100% of the G channel of ***Coat material #1***. This is because for this channel the ***Blend Amount*** value is 255. Lastly, for the B (Blue) channel of the final material, 100% of the Base material is used, just like the R channel. This is because the ***Blend amount*** value of the B channel is equivalent to 0.



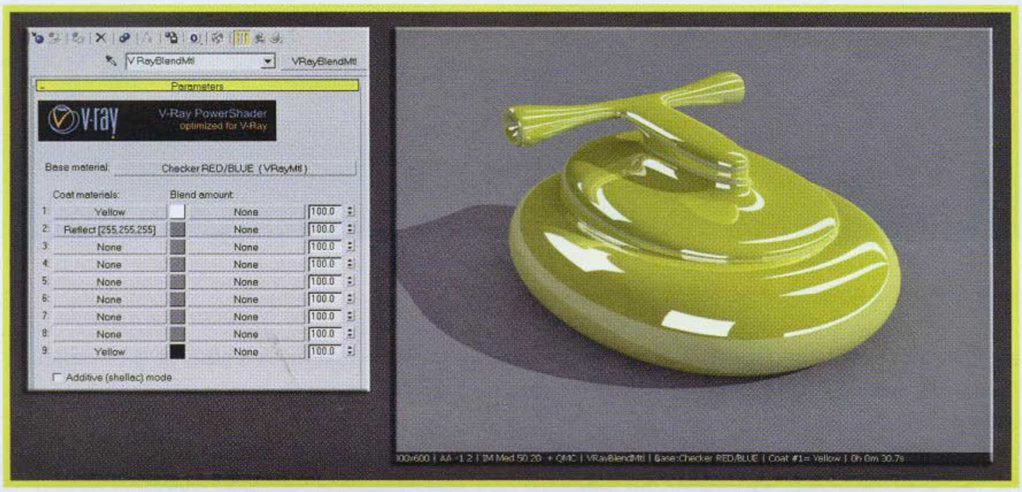
■ **Figure 5.163**
Blend amount: graphic representation of its function via RGB colors.

Obviously this is the simplest case, where only the ***Diffuse*** channel is taken into consideration. One can however, use more than one layer. Their contribution can for example be used only for reflections.



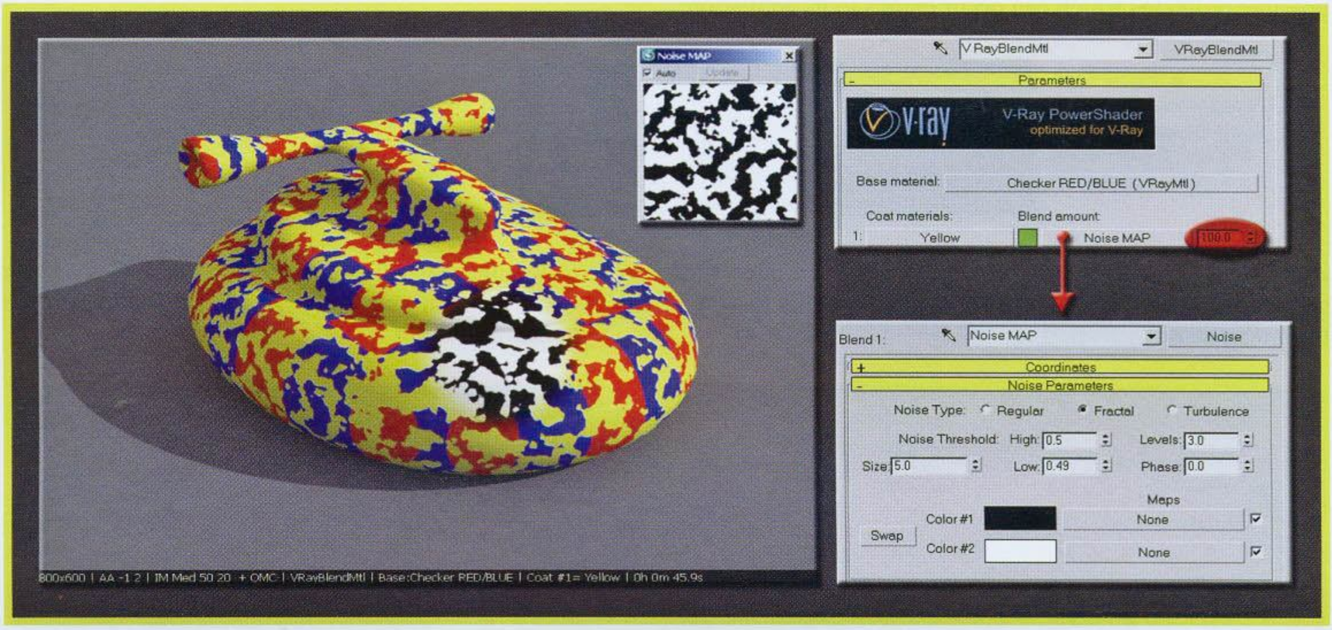
■ **Figure 5.164**
A second layer being added.

Before analyzing the complete material, observe the ***Coat material #2***. It is a completely reflective material with ***Reflect color*** RGB = [255, 255, 255] and black ***Diffuse*** color. The final material is not completely reflective. This is because the color of ***Blend amount*** of the ***Coat material #2*** is RGB = [128, 128, 128]. Basically in the final result the ***Coat material #2*** contributes to reflection by 50%. In the following image the final material has a ***Diffuse*** of 50% of RGB = [255, 255, 0], ***Coat material #1*** is 50% of RGB = [0, 0, 0], the ***Coat material #2***. It will also have 50% reflection of ***Coat material #2***. Base material has not had any effect because the color of the ***Blend amount*** of ***Coat material #1*** is pure white.



■ **Figure 5.165**
White ***Color Blend*** amount. In this case all layers above ***Coat material #2*** are blocked.

One can also use a map, colored, black or white, allowing one to define both the position and application intensity of layers.



■ **Figure 5.166**
Blending carried out with a black and white map.

In the previous material there is an active texture in the ***Blend amount*** channel, with the precise aim of determining areas and blending intensity of the layer. The green color of ***Color Blend amount*** does not seem to have any effect; in fact the colors of ***Base material*** and ***Coat material #1*** are the original ones. By paying closer attention we notice that the spinner on the right-hand side of the ***Noise*** map displays the number 100. This means that it is 100% of the ***Noise*** map which decides blending. Another observation concerns the way the ***Noise*** map intervenes on the blending of the two maps. The completely black areas allow the rendering of ***Base material***, whereas the white ones allow the rendering of ***Coat material #1***.

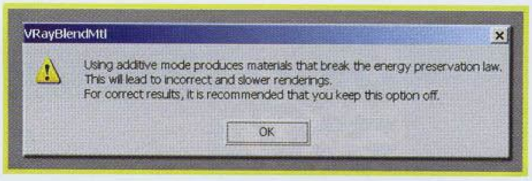
By changing the colors of the ***Noise*** map, the whole material changes color. By changing the spinner next to the map, ***VRay*** uses both the blending map and the ***Blend amount*** proportionally to the spinner for blending the layers.

■ **Figure 5.167**
Modification of the ***Blend amount*** spinner. The level of complexity of the material increases.

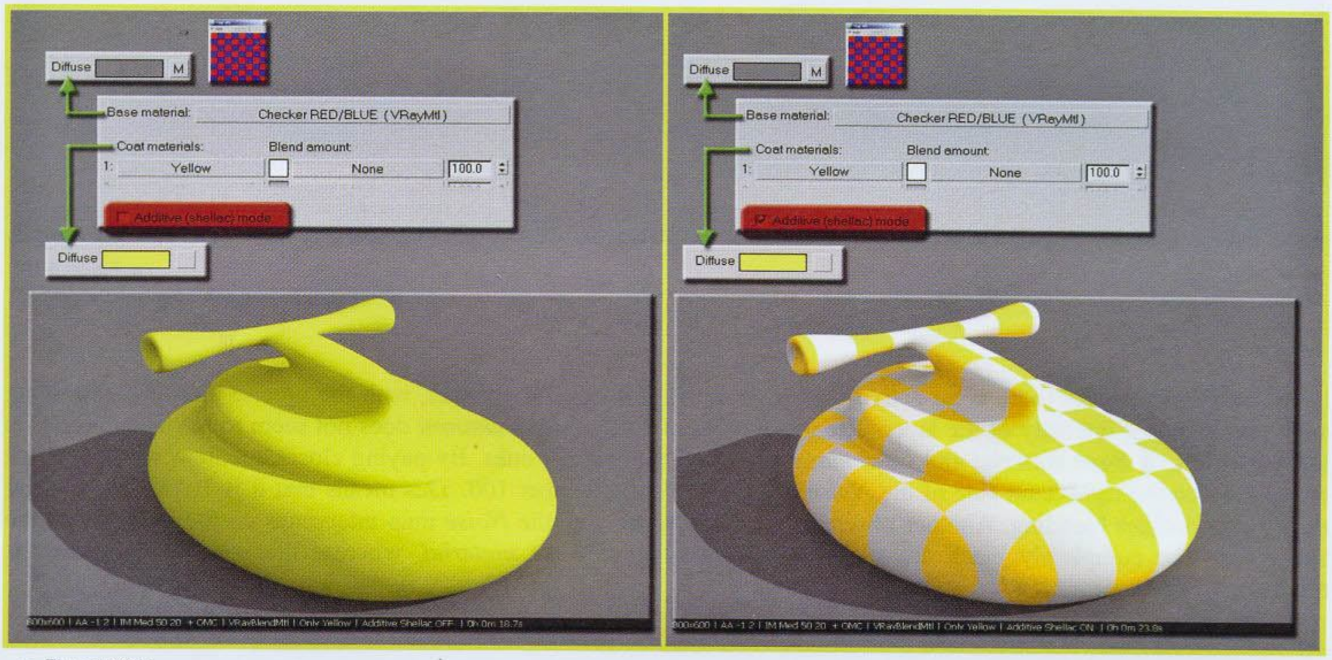
Changing the spinner adds another parameter for modifying blending between the two layers. A value of 0 completely nullifies the effects of the ***Noise*** map, whereas a value of 100 does so for color ***Blend amount***. Intermediate values mix the effects, giving a combination of the two levels.

Additive (shellac) mode - by activating this parameter the behavior of ***VRayBlendMtl*** becomes identical to that of the 3ds Max ***Shellac***. One must remember how this last material can sometimes generate results which are not physically correct, such as a material with ***Diffuse*** color and 100% reflection, a situation which is impossible in real life. This parameter must therefore be used with care. By activating this option a window appears warning the user of possible problems which can occur thanks to its activation.

■ **Figure 5.168**
The warning generated by VRay due to the activation of the ***Additive (shellac) mode*** parameter.

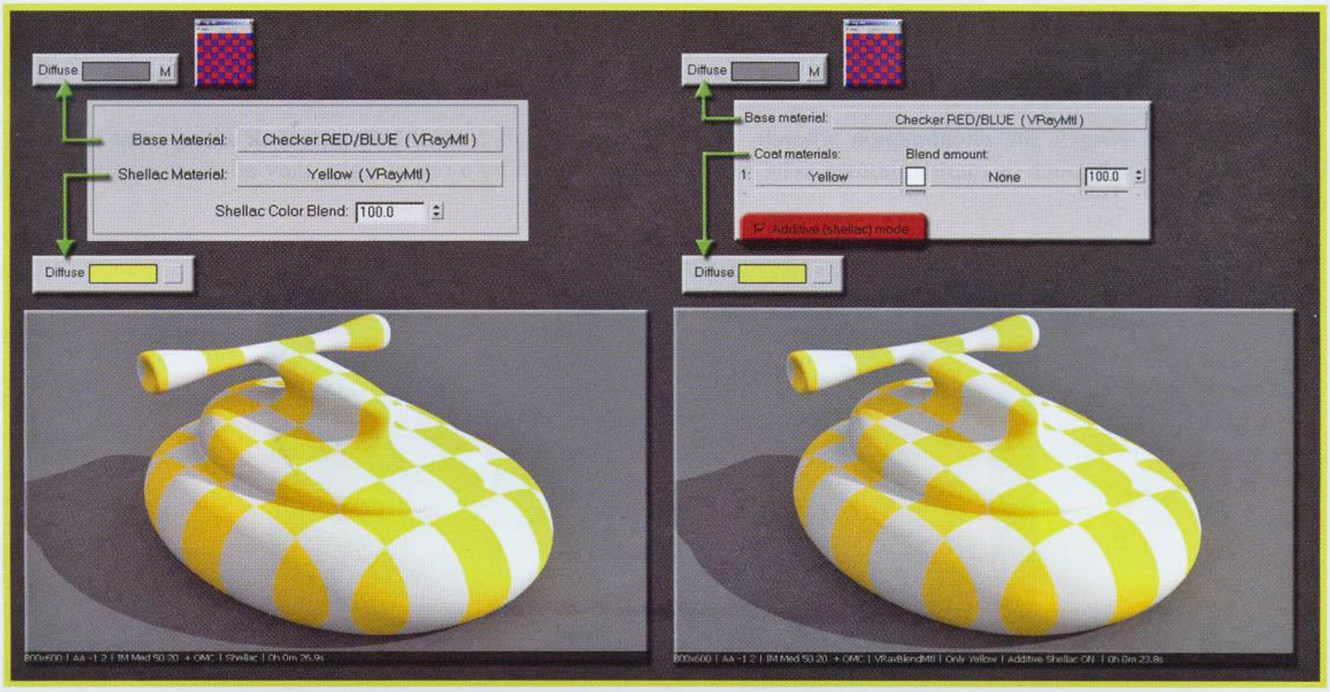


By using this mode one may risk creating materials which do not follow the law of energy preservation. This in turn may cause an incorrect and slow render. For this reason, unless one knows what one is doing, it is advisable to leave this parameter deactivated.



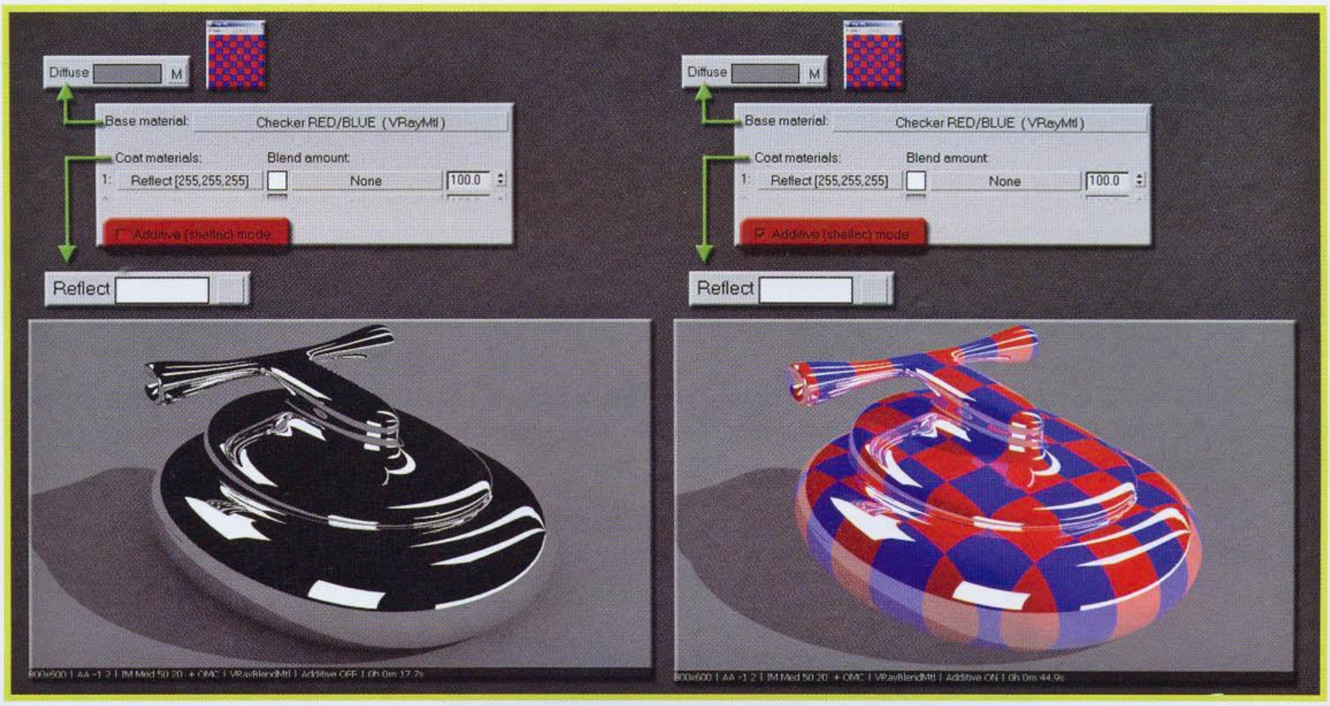
■ **Figure 5.169**
Activation and deactivation of the ***Additive (shellac) mode***.

In the left-hand image, the one with the ***Additive (shellac) mode*** parameter deactivated, the result shows what should happen correctly. The ***Blend amount*** color is perfectly white and only ***Coat material #1*** is visible. By activating the ***Additive (shellac) mode*** parameter the result is surprisingly identical to the one obtained with the ***Shellac*** material. The single layers have simply been summed, regardless of the principle of the preservation of energy.



■ **Figure 5.170**
Comparison between the 3ds Max ***Shellac*** and the ***VRayBlendMtl*** of VRay. The result is identical.

Another example can be described via the use of a ***Coat material #1***, dedicated exclusively to reflection.



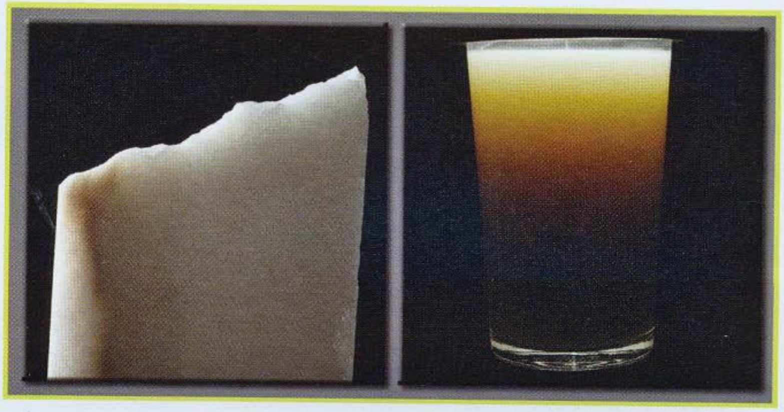
■ Figure 5.171
Coat material #1: reflection only.

In the left-hand image, ***Coat material #1*** contains a ***VRayMtl*** with a reflection channel RGB = [255, 255, 255]. The ***Color Blend*** amount is completely white, which makes the material 100% reflective. By activating the ***Additive (shellac) mode*** parameter one can see how in the right-hand image the final material changes its appearance completely. It has been acquired with a material with 100% ***Base material*** and 100% reflection, a situation which is not possible in nature.

VRay Material : VRayFastSSS

INTRODUCTION

Subsurface scattering is the phenomenon whereby light travels through a semi-transparent or translucent material. When these materials, such as marble, human skin, wax and so on are hit by a ray of light, they allow energy to spread within them.

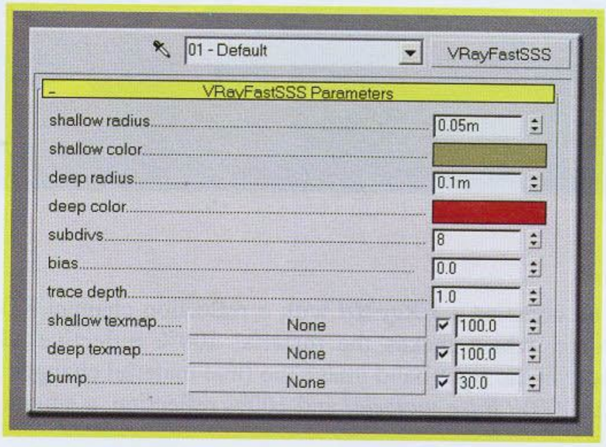


■ **Figure 5.172**
Some examples of ***Subsurface scattering***.

Subsurface scattering has already been amply covered in the section concerning ***VRayMtl*** relative to translucency. So before continuing with this explanation of ***VRayFastSSS***, it is advisable to have carefully read that chapter first. ***VRayMtl*** already has the ability to simulate ***SSS*** and there are two calculation modes: a non-interpolated unbiased one and a biased one. Obviously the translucency calculation via the direct method with the ***QMC*** method requires time and resources. This is why ***VRayFastSSS*** has been introduced, a new unbiased system for calculating translucency. This method is called Fast simply because it uses a simplified version of the calculation algorithm used in the ***VRayMtl***. This does not allow one to obtain the same quality as ***VRayMtl***, but calculation time is much less and the results are high quality. The setting of parameters is also much simpler, as they are limited and easy to understand.

PARAMETERS

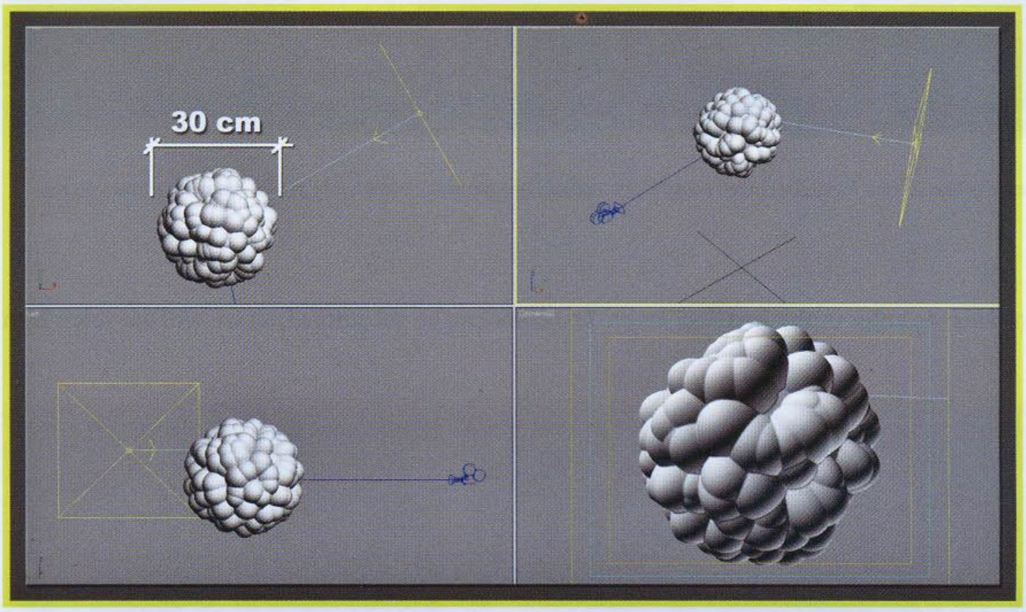
VRayFastSSS simulates ***SSS*** with the use of two layers. The first of these handles translucency in the external zones of the model, the second takes care of the internal zones, directly underneath the first layer. It is possible to manage the depth of the effect, the color and the quality.



■ **Figure 5.173**
VRayFastSSS material.

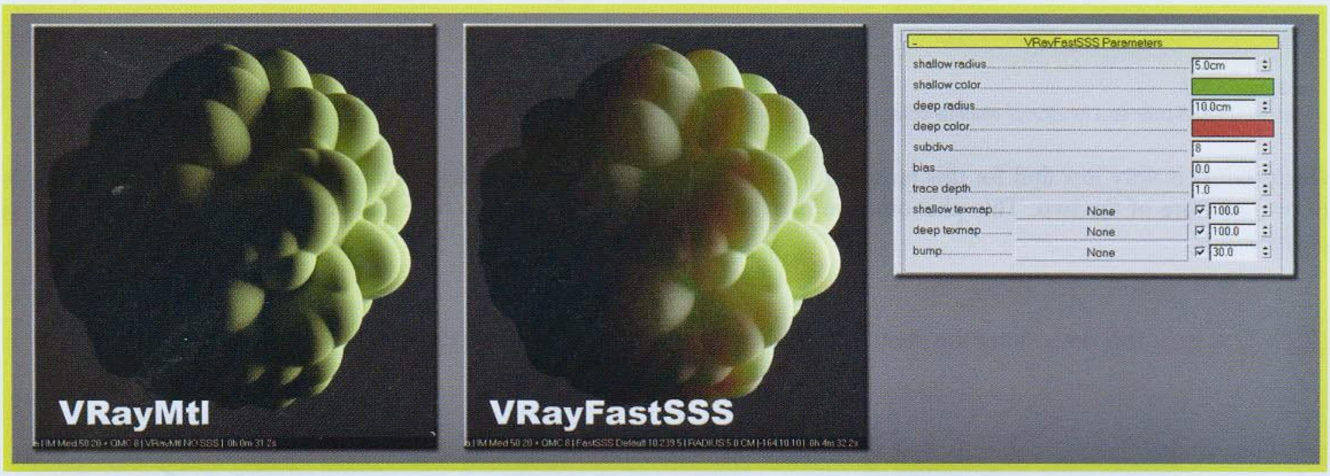
shallow radius - this parameter determines the radius, that is, the depth, of the first level of translucency. The higher this value is, the more the color of the first layer acts deeply. For better results it is advisable to keep this value below ***deep radius***, a parameter that determines the depth of the second layer. In the opposite case, the first layer would cover the effects of the second one, as it would intervene more deeply.

The scene used for the test is made up of sphere which has been changed geometrically, illuminated from one side by a ***VRayLight*** source.



■ **Figure 5.174**
The scene used for the test on ***VRayFastSSS*** material.

With a simple material, the result is very flat, whereas with a ***VRayFastSSS*** material and default settings, the result changes.



■ **Figure 5.175**
Two materials being compared. A ***VRayMtl*** without ***SSS*** and ***VRayFastSSS***. The ***VRayMtl*** only has one color in the ***Diffuse*** channel. As far as ***VRayFastSSS*** is concerned soon we will see how to improve its efficiency.
--- DVD ---

The ***VRayFastSSS*** used in this image has default parameters, except for colors. Notice how the size of models influences the material. The sphere has a diameter of 30cm and the depth of the first level is equivalent to 5 cm. This means that the first layer affects a maximum depth of 5 cm. The second level instead reaches 10 cm. In the render, after a certain depth, one can start to see the reddish color of the second layer.

■ **Figure 5.176**
With a depth of 30 cm, the influence of the second level disappears completely, making the geometry completely green. Rendering times do not vary much.
- - - DVD - - -

shallow color - this determines the color of the first ***SSS*** layer. The color can be substituted, as we will see later, by a map.

■ **Figure 5.177**
Variation of ***SSS*** color as a function of ***shallow color***.

Deep radius is very low and for this reason it is difficult to distinguish the colors of the first two levels clearly. It tends to be a mixture of the two. This phenomenon is evident in the first test where blue and red unite and give the sphere a purple color.

deep radius - this determines the depth of action of the second ***Subsurface*** layer.

■ **Figure 5.178**
Variation of the ***deep radius*** parameter. Now ***SSS*** is more visible, making translucency efficiency better.
- - - DVD - - -

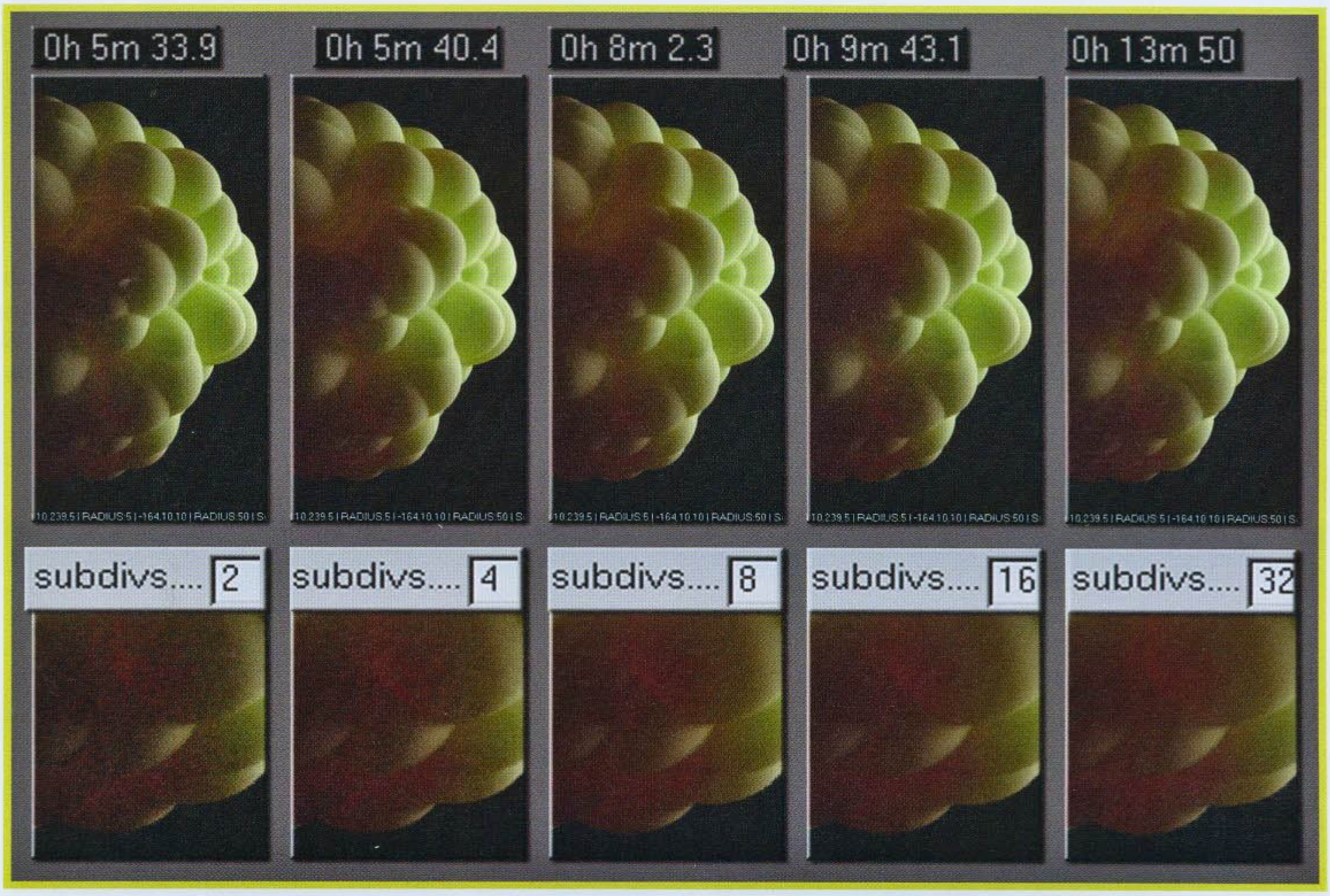
The ***deep radius*** parameter represents the second layer of the ***SSS***. It allows one to define the depth of ***SSS*** and improve the diffusion of light. This however has a price which is rendering time, which is practically doubled. Now we can see the back of the geometry, illuminated by the light which has spread within the sphere.

deep color - this represents the color of the second layer. As we will see soon, the simple color of the level can be substituted by a texture.

■ **Figure 5.179**
Variation of color of the second ***SSS*** layer.

By modifying ***deep color***, the deepest part of the geometry changes color. Looking at the images, the green of the first layer, the superficial one, is visible in any case in the right-hand side of the model directly illuminated by ***VRayLight***. The other zones, also thanks to a high level of ***deep radius***, are completely texturized from the second layer.

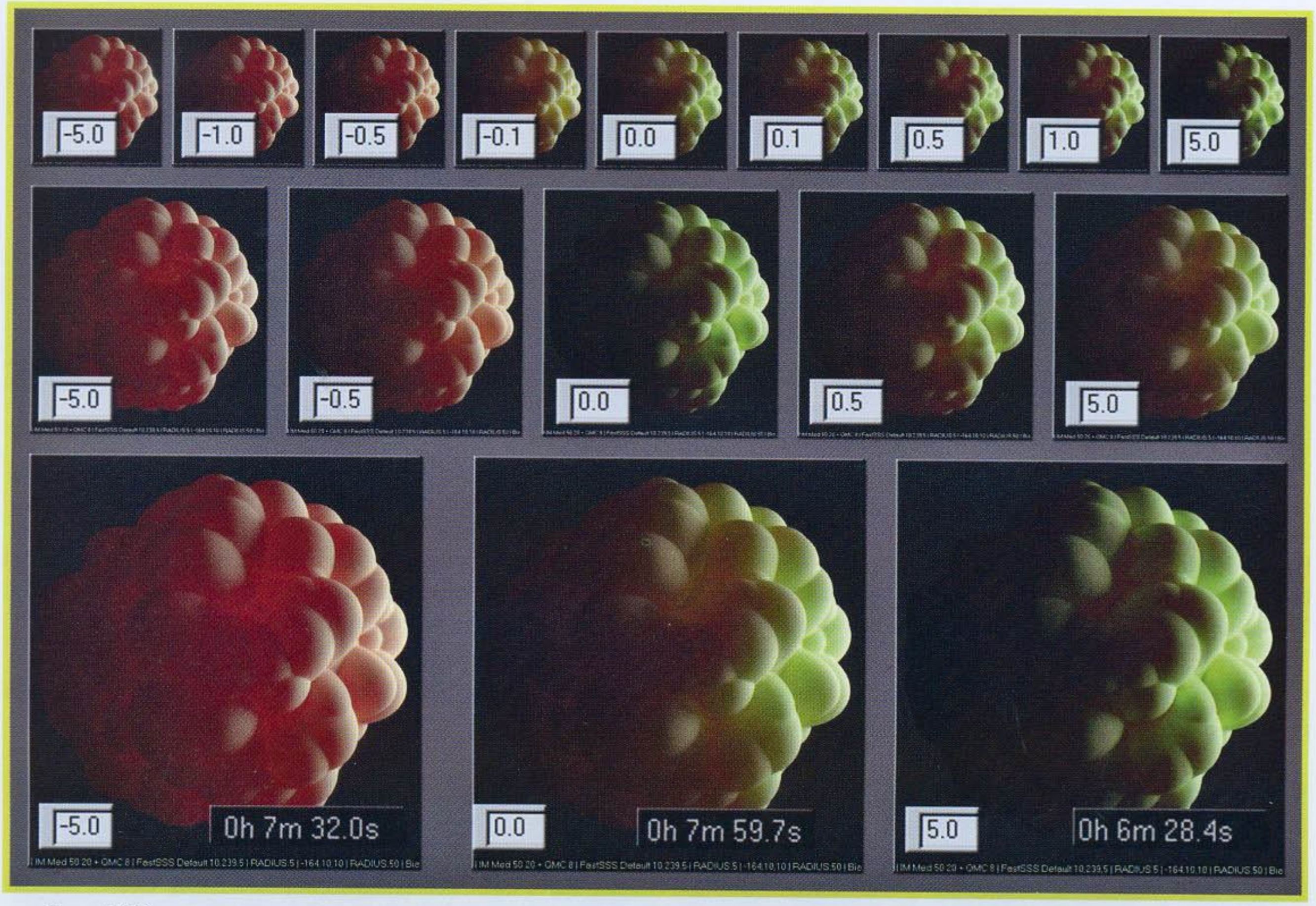
subdivs - this determines the number of samples and therefore the quality of ***Subsurface Scattering***. High values produce high quality renders without noise but rendering time increases.



■ **Figure 5.180**
Different quality levels. Rendering time increases.
--- DVD ---

Like all the parameters of *VRay* managed by ***QMC***, also ***VRayFastSSS*** has the ***subdivs*** parameter. This parameter improves rendering quality at the expense of time. The default value allows one to obtain a good quality render, but during the production phase it is advisable to keep ***subdivs***= 16/32. Noise is present in the more difficult areas like shadowy ones.

bias - this controls the management of ***scattering***. High values determine a lower dispersion of light.



■ **Figure 5.181**
Different settings for the ***bias*** parameter.

What does less dispersion actually mean? It means energy given to the light which penetrates within the model, which basically means less ***SSS***. If one applies a very low ***bias***, the sphere becomes very red. This is because light penetrates deeper, where the influence of ***deep radius***, which is red, is greater. Vice versa, with a very high ***bias***, the effect of ***SSS*** tends to diminish, because light has less energy. Consequently the mesh is less translucent and shows only the shallow layer.

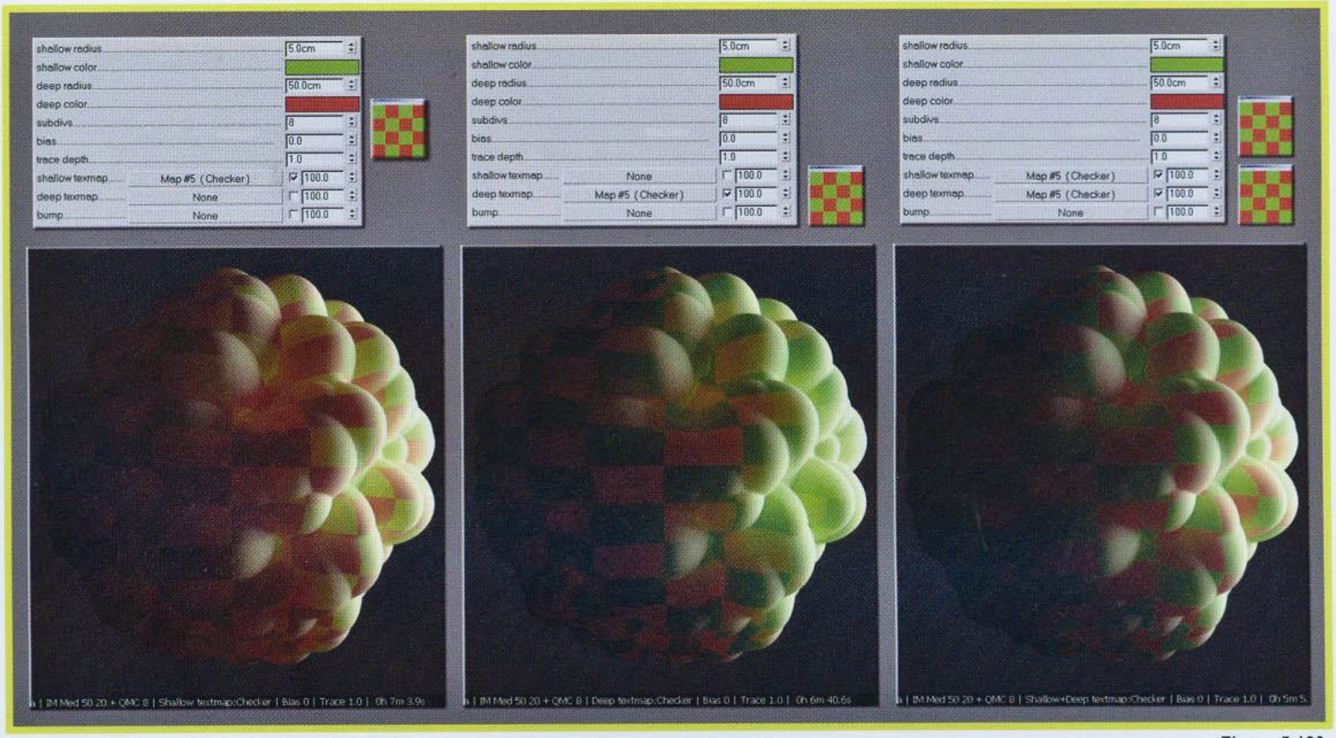
trace depth - parameter used by *VRay* for deciding the length of rays used for the calculation of ***SSS***.

■ **Figure 5.182**
Variation of the ***trace depth*** parameter.

This parameter limits the depth of calculation of ***Subsurface Scattering***. By analyzing renders, with ***trace depth*** = 0.01 it is possible to observe the fact that inner zones of the geometry, those which should be red, are instead black. This happens because the calculation level of translucency has been limited only for the superficial parts which are managed by the green ***shallow*** layer. By increasing the ***trace depth*** value, depth increases and with it also the contribution of ***deep layer***. Lastly, if strange artifacts should appear in the ***SSS***, lowering this parameter could be useful as it would mean limiting the effect itself.

shallow texmap - by inserting any map, one can manage the color of the first ***SSS*** layer by means of textures. With the numerical spinner one can decide the percentage of influence on ***shallow color***.

deep texmap - by inserting any map, one can manage the color of the second ***SSS*** layer by means of textures. With the numerical spinner one can decide the percentage of influence on ***deep color***.



■ **Figure 5.183**
A Checker map being used for defining both the first and second layer.

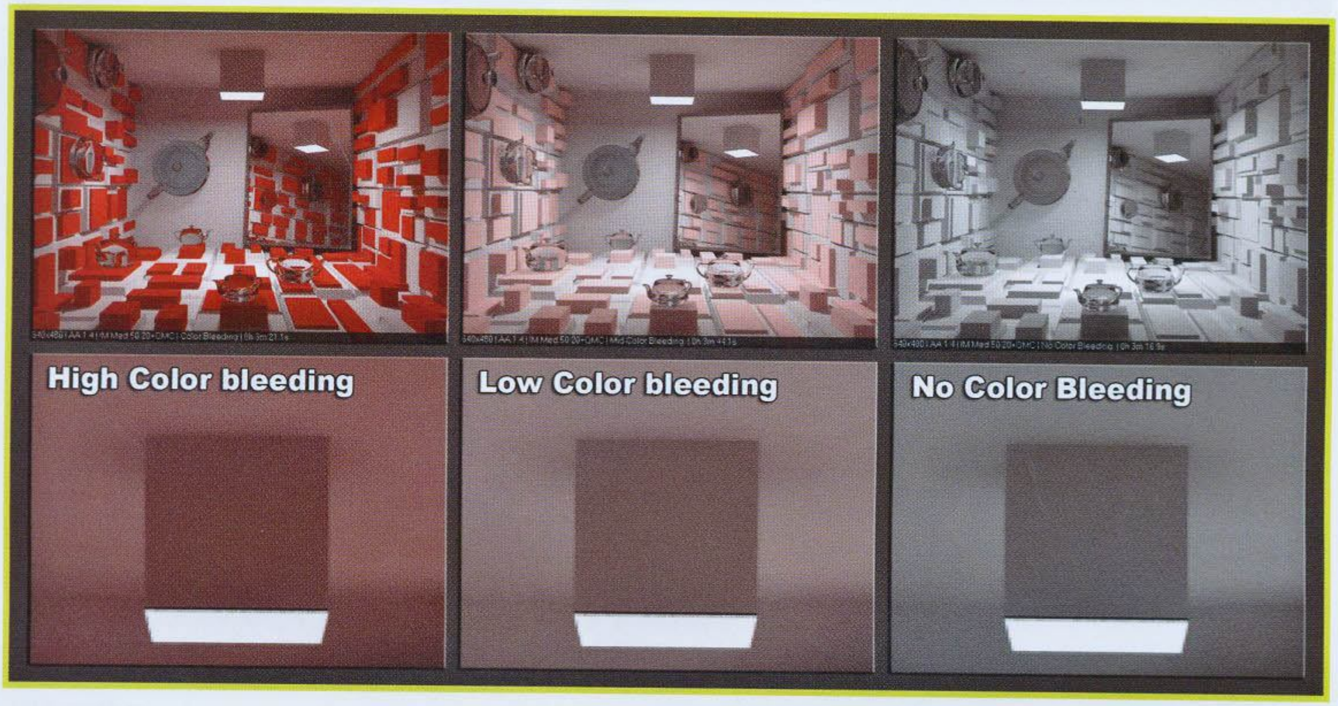
By using the Checker map as a texture for the first level, the green tone in the Color Selector will not be used anymore, but in its place the procedural map. By clicking on the small checkbox on the right, the map is activated and with the numerical spinner one can define the blending of color and texture. The same process takes place for ***deep layer***. It is possible to employ them together.

Bump - by inserting a map in black and white, one can generate the Bump effect as well as ***SSS***.

VRay Material : VRayOverrideMtl

INTRODUCTION

This particular material, as well as the ***VRayWrapperMtl***, can be defined as a "utility" material, as it allows one to change the characteristics, such as the generation of Global Illumination, reflection and refraction of any material, as well as indirectly controlling also Color Bleeding. The term Color Bleeding is the name given to the phenomenon by which color is spread due to GI action on a colored or texturized material, which diffuses its color to nearby geometries.

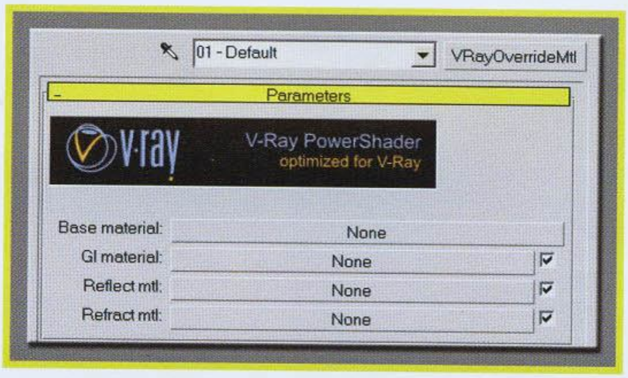


■ **Figure 5.184**
The effect of color on ***Color Bleeding***.

The red color of the blocks lit by the light source spreads to the walls which are white. The result is that they take on a slight pink tone. With ***VRayOverrideMtl*** one can control the intensity of color diffusion for instance.

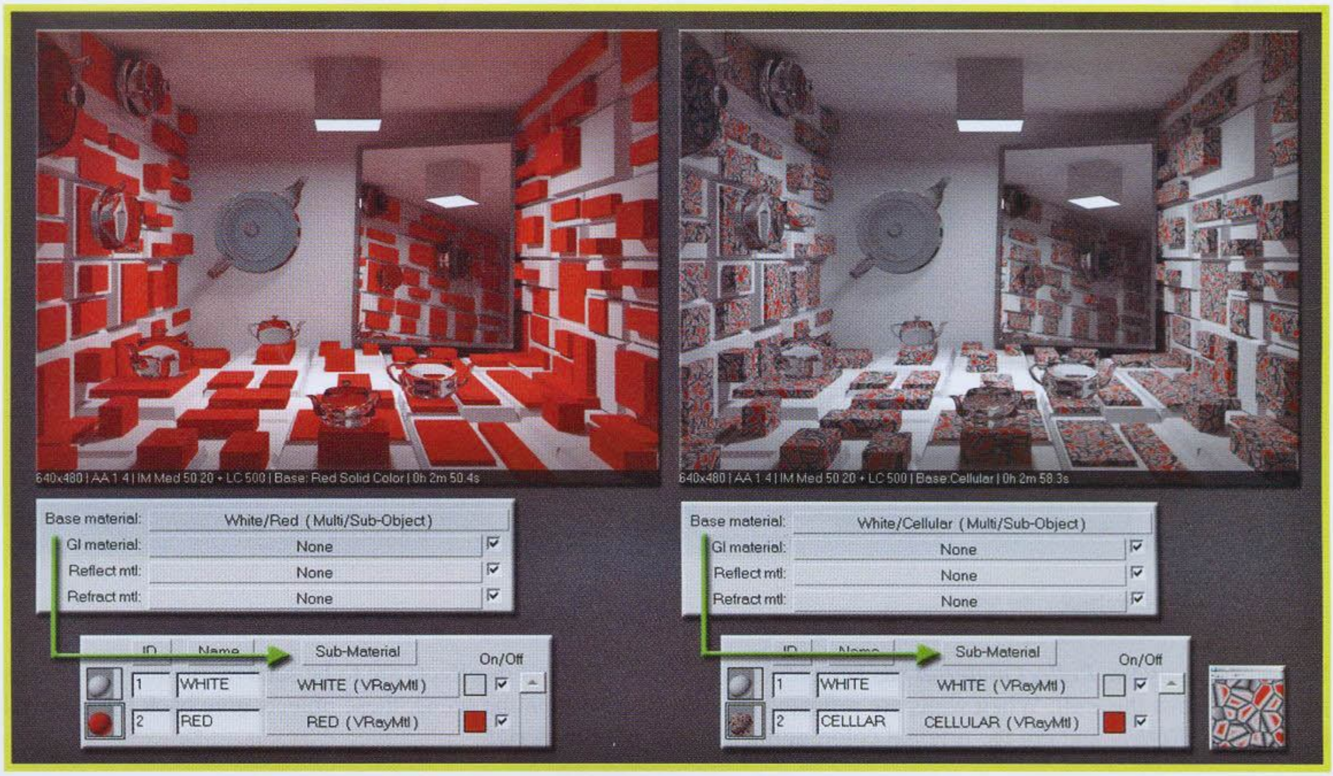
PARAMETERS

■ **Figure 5.185**
The ***VRayOverrideMtl*** material.



Like the ***VRayWrapperMtl*** and ***VRayBlend*** materials, the first thing that needs to be done is to specify the ***Base material***, the main material which will be used for the following variations. In the case we are examining we will use a ***VRayMtl***. The three remaining slots allow one to specify, by means of the same number of materials, the shaders to be used by ***VRay*** for the calculation of Global Illumination and the calculation of reflections and refractions in substitution of the ***Base material***.

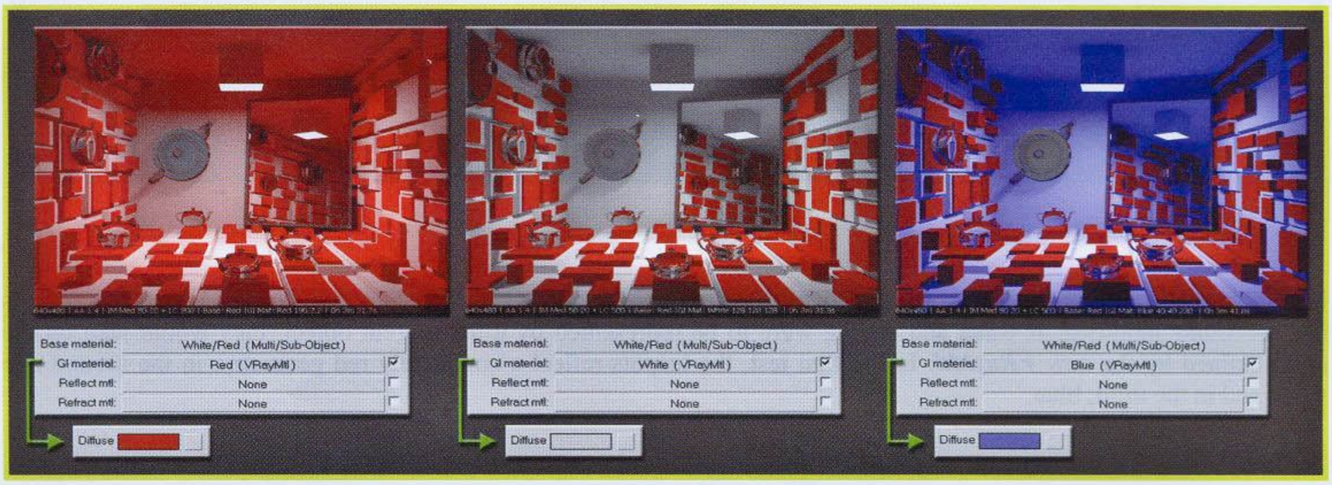
Base material - this represents the base material, which is used for the actual render.



■ **Figure 5.186**
Two different ***Base materials*** being used.

In the previous tests two different ***Base materials*** have been used. In both cases a Multi/Sub-Object has been used because a new ID is assigned to the geometries of the model, which are generated by the Greeble plug-in. The small boxes have ID #2 and the remaining parts instead have ID #1.
In the left-hand test a simple red ***VRayMtl*** has been assigned to the parallelepipeds, whereas the one on the right is also a ***VRayMtl*** but with a Cellular procedural map in the ***Diffuse*** channel. Notice the difference of Color Bleeding on the ceiling.

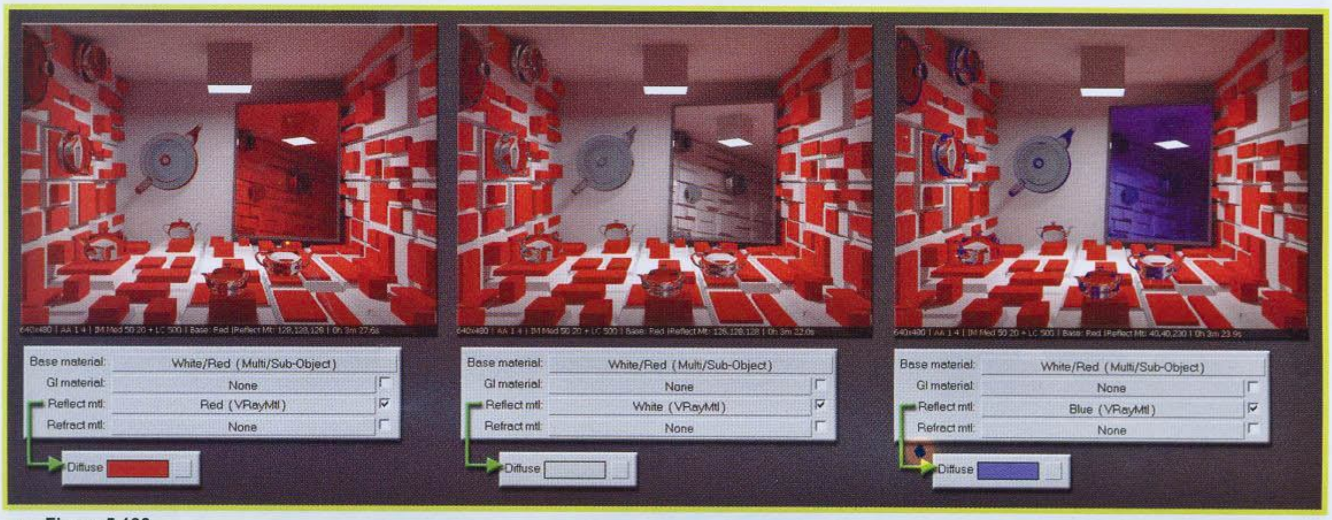
GI material - for the calculation of Global Illumination alone, instead of the ***Base material*** VRay uses the material found in this slot. It is a very efficient system for reducing the effects of Color Bleeding which are too evident.



■ **Figure 5.187**
Different use of ***GI material***.

A material applied in this case to the walls changes the *VRay*'s behavior and the calculation of the GI. By using a red material, the final result changes completely and saturates the GI. Instead by using a grey ***VRayMtl***, the result is the complete elimination of Color Bleeding, neutralized by the grey tone. Lastly, one can completely modify the look of the render by using very vivid colors. Like in the last case, where a bright blue tone has been used. From the physical point of view, this type of result is not correct, but visually speaking there can be cases where one has the necessity to generate GIs of a different color from the original ***Diffuse***.

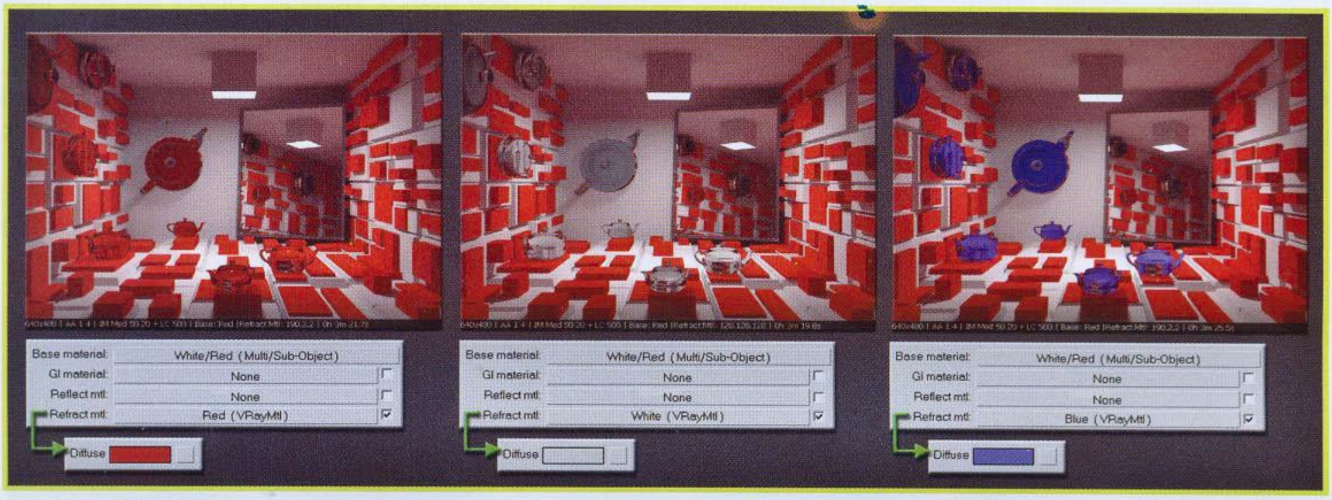
Reflect material - *VRay*, for the calculation of reflections, uses a material found in this slot in the place of ***Base material***.



■ **Figure 5.188**
The effects of ***Reflect material*** on the render.

In this case, only the ***Reflect material*** has been replaced with a color and its effects are to be seen in the mirror which is standing against the wall. When walls are reflected by chrome objects in the scene, these objects give back the material inserted into ***Reflect*** and not the original color, defined by the ***Base material***.

Refract material - *VRay*, for the calculation of refractions, uses the material in this slot in substitution of ***Base material***.



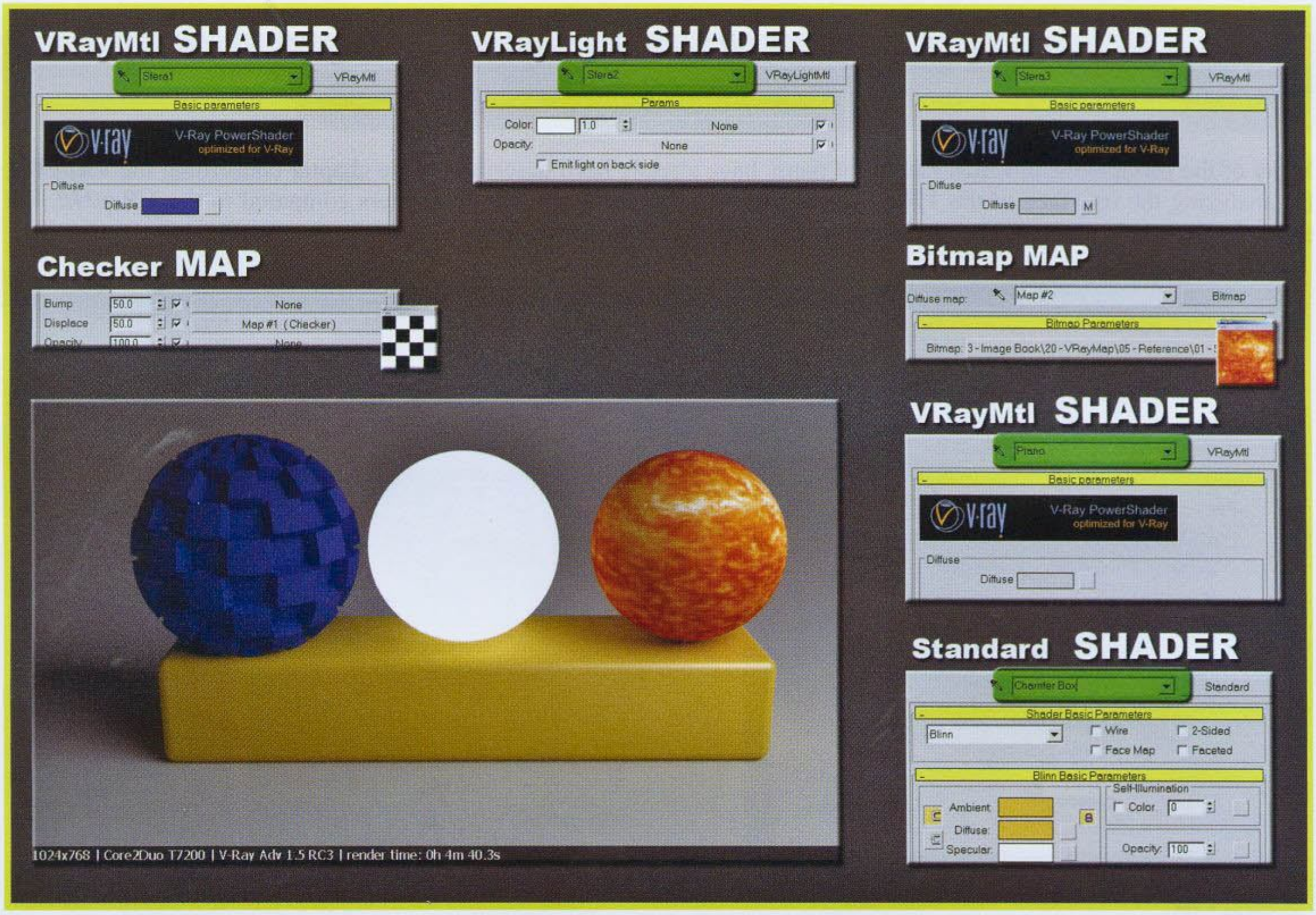
■ **Figure 5.189**
The effects of ***Refract material*** on the render.

In this case, ***VRayOverrideMtl*** is associated to the ***Reflect Mtl*** channel occupied by a ***VRayMtl*** which has had its ***Diffuse Color*** modified. If walls and ceiling were to be represented by a transparent object, they would take on the color defined by the new ***VRayMtl***. In the first test, the teapots reflect a completely red geometry, in the second one white and in the third, blue.

VRay Map

INTRODUCTION

A material, by definition, is an element which allows one to define the way a 3D object reflects other geometries, light transmission and so on. There are also other materials which allow one to combine multiple shaders and others which increase their properties, like ***VRayOverrideMtl*** which we have seen in the previous paragraph. Within materials one can use maps which allow the use of textures, reflections, refractions, ***HDRI*** images and other effects which will be analyzed shortly.
A map is one of the building blocks that allows one to create materials. It is possible to use maps in the place of simple color or black and white maps to define the quantity and position of reflection, etc...



■ **Figure 5.190**
Different results obtained with the use of different materials and maps.

In this scene five different types of material with unique properties have been used. The first sphere, for instance, has had a ***VRay*** material with a 2D Checker map assigned to it, which has contributed towards the definition of ***Displacement***. A ***VRayLightMtl*** has been assigned to the second sphere in order to give the geometry luminosity. In the last sphere, the ***Diffuse*** channel is managed by a Bitmap texture. The plane is a simple ***VRayMtl*** and the parallelepiped which it rests on is a Standard material, the default shader in 3ds Max.
It is clear that textures can be used to improve or modify certain aspects of materials. In the next pages we will analyze the maps made available in ***VRay***.

VRay Map: VRayMap

INTRODUCTION

VRay is a rendering engine of the raytrace type. So is 3ds Max and both of them use proprietary algorithms for rendering. For this reason, when using a Raytrace material or map, using *VRay* as the rendering engine would almost certainly lead to calculation and visualization problems. This is because *VRay* is not able to "read" these types of materials properly as they are exclusively for 3ds Max. Therefore it is unadvisable to use them.

By using Scanline to create chrome or glass materials with the Standard material of 3ds Max, it was necessary to assign a Raytrace map in the ***Reflection*** or ***Refraction*** channels. These maps however are not supported by *VRay*. Thanks to the ***VRayMap*** it is in any case possible to render reflective or transparent materials with also with the Standard material with *VRay*. The only operation that needs to be carried out is to use the ***VRayMap*** in the place of the 3ds Max Raytrace maps by inserting it in the ***Reflection*** or ***Refraction*** channels in the Standard material.

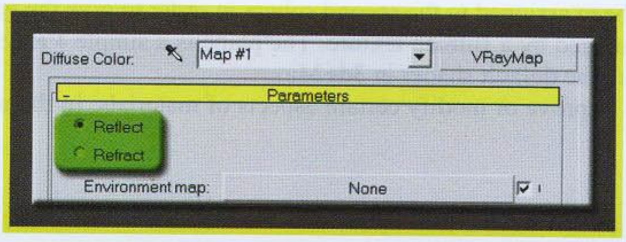
PARAMETERS

Many of the parameters, in actual fact all of them, have already been described in the chapter concerning the ***VRayMtl***. By analyzing the interface of the ***VRayMap*** one notices that there are less parameters compared to the ***VRayMtl***. For example, the whole part concerning reflection interpolation. Also translucency is missing.

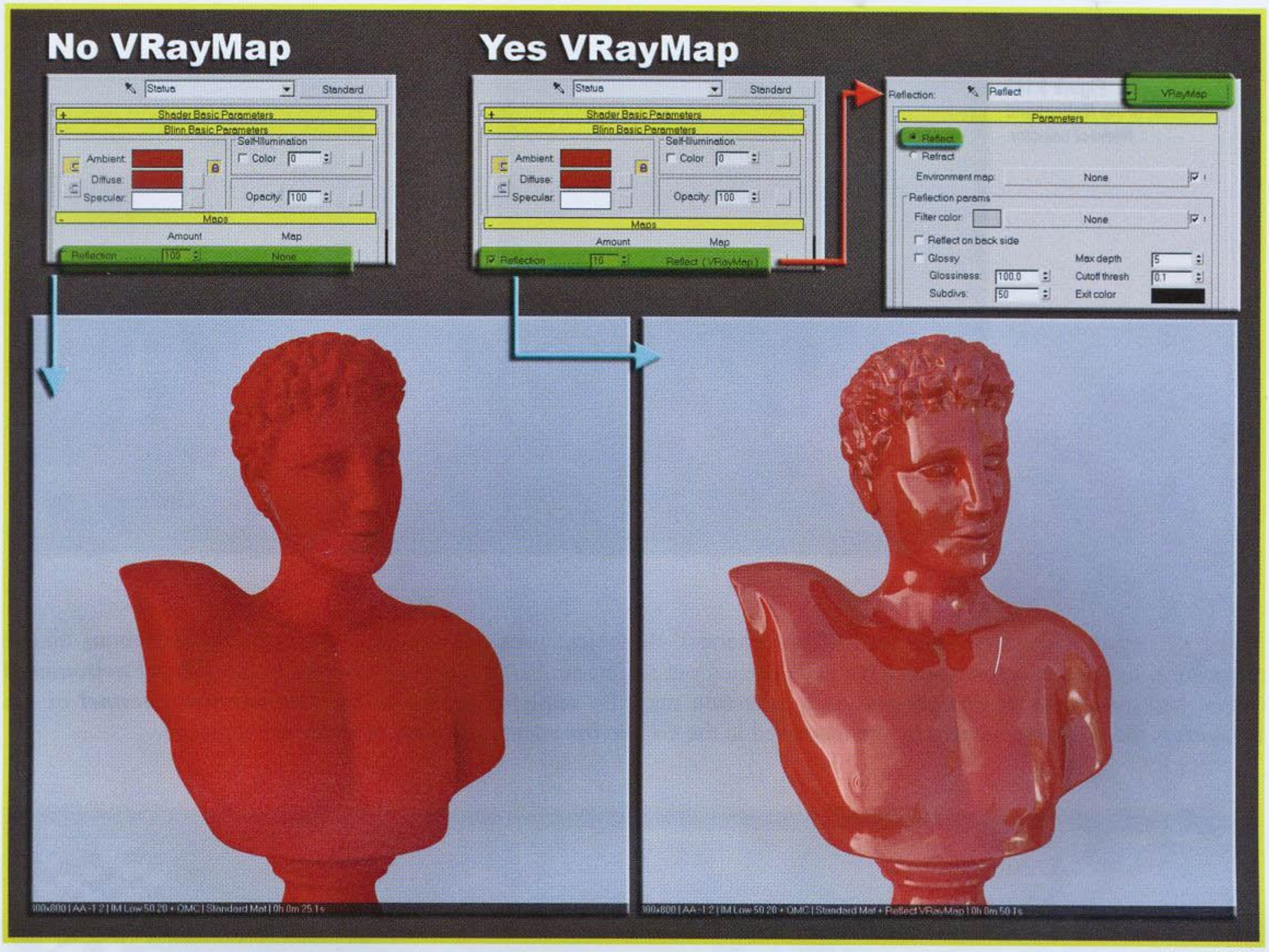
■ Figure 5.191
The *VRayMap*.

The ***VRayMap*** is made up of two areas: one dedicated to reflection and one to refraction. According to its adjustment in the Standard material in the ***Reflection*** and ***Refraction*** channels, the relevant top left option-box in the interface must be checked. This defines reflective or refractive functions.

■ Figure 5.192
The *Reflect* and *Refract* selector.

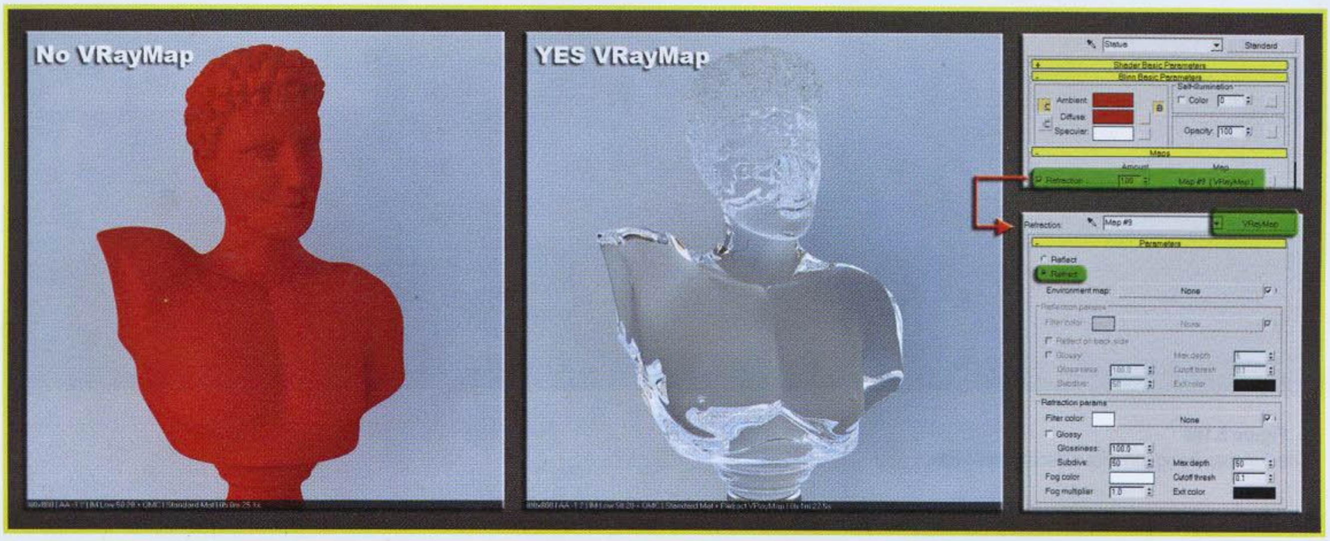


If the purpose is that of creating a chromed material, one must place the ***VRayMap*** in the ***Reflection*** channel and check the ***Reflect*** selector located in the ***VRayMap***. This way ***VRay*** knows that the ***VRayMap*** must reflect.



■ **Figure 5.193**
The ***VRayMap*** being used in the reflection channel.

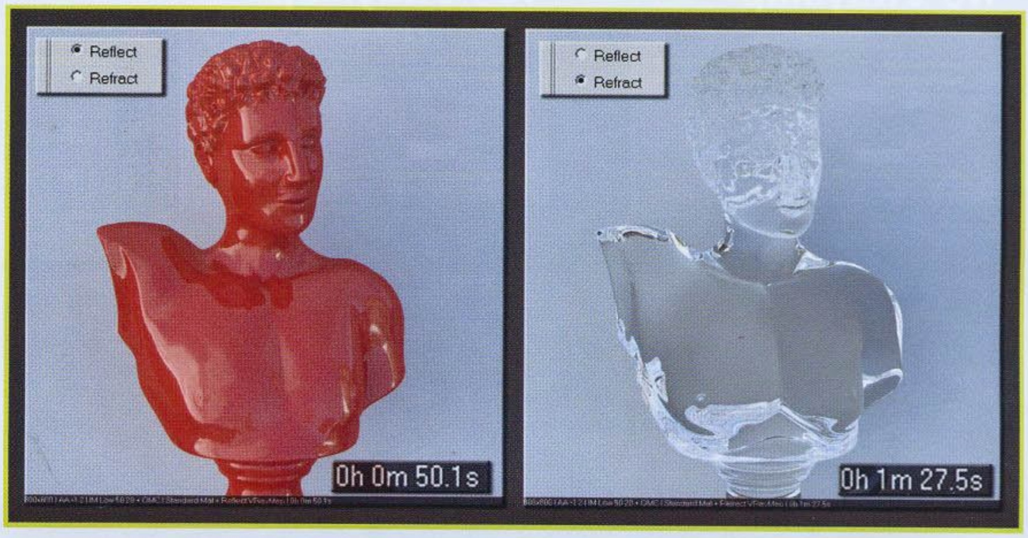
In the left-hand test a Standard red material has been used. In the right-hand one, the same material has had a ***VRayMap***, with ***Reflect*** option activated, added to the ***Reflect channel***. This means that the section of the ***VRayMap*** concerning refraction is grayed out and one can only alter reflection settings. In the test default values have been used, except for ***Filter color***, changed to RGB = [128, 128, 128]. Lastly the ***Amount*** setting of the ***Reflection channel*** has been lowered to 10. This is done to avoid total reflection. In order to obtain a glass object, it is enough to insert a ***VRayMap*** with the selector set to ***Refract*** in the ***Refraction*** channel.



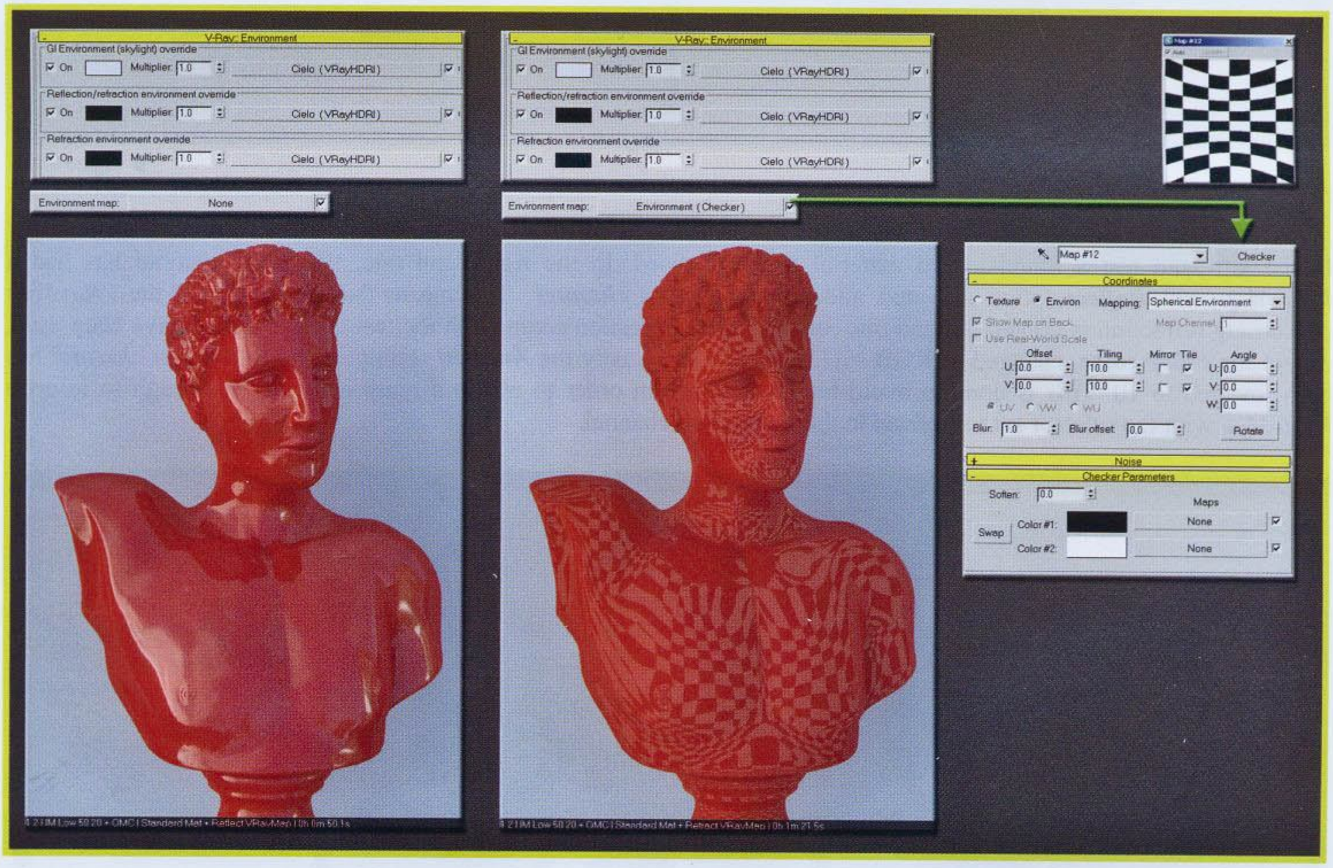
■ **Figure 5.194**
The ***VRayMap*** being used in the refraction channel.

Selector - it is possible to manually determine the use of the ***VRayMap***. If the purpose is to use it for reflection, one must insert it in the ***Reflection*** channel of the 3ds Max Standard material and select ***Reflect***. If one wishes to use it for refraction, one must insert it in the ***Refraction*** channel of the 3ds Max Standard material and select ***Refract***.

■ **Figure 5.195**
Variation of the ***Reflect*** and ***Refract*** selector.



Environment - by inserting a map in this channel, the object uses this texture for generating reflections and/or refractions. In the examples which have been carried out so far, an ***HDRI*** has been inserted in the ***VRay:Environment*** rollout and used both as ***skylight*** and as a reflection map. By applying a map to the ***Environment channel*** of the ***VRayMap***, the effects generated by the map used in the ***GI Environment (skylight) override*** are replaced.



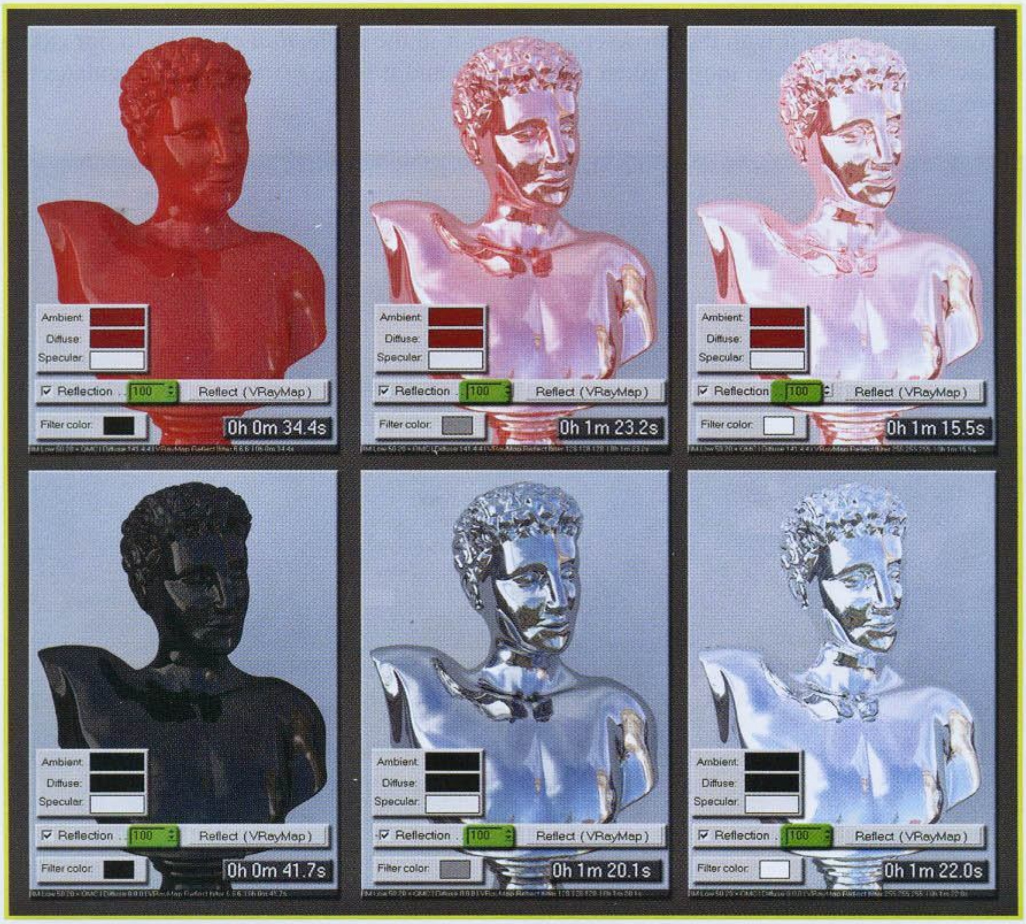
■ **Figure 5.196**
Checker map being used for reflection definition.

Reflection params

The parameters in this section are identical to those covered in the analysis of the ***VRayMtl***. Therefore there is no need to go over them a second time. For an in-depth explanation one can read the chapter concerning ***VRayMtl***. Only two parameters are slightly different. The first one is ***Filter color***, which in ***VRayMtl*** is called ***Reflect color***. The second one is ***Glossiness***. In ***VRayMtl*** this parameter varies from 0 to 1. With value = 1 reflection is not blurred, whereas from 1 downwards blurring increases. When one uses the ***VRayMap***, settings lower than 100 increase the amount of blurring, settings higher than 100 lower it. For all the rest, there is no difference compared to the ***VRayMtl***, although the ***VRayMap*** does not have interesting features such as the possibility to interpolate reflection or define reflection IOR. To obtain ***Fresnel*** it is enough to insert the ***VRayMap*** into a Falloff map, therefore the fact that it is missing can be solved this way and ***Highlights*** can be created with the options located in the ***Basic parameter*** rollout of the Standard material. Notice how default value for ***Subdivisions*** is particularly high. This has perhaps been overlooked by the programmers. Until a few versions ago in fact, also ***Subdivisions*** in ***VRayMtl*** were by default set to 50, but this value was in most cases too high. Even with ***Subdivisions*** set to 8 one can obtain decent quality.

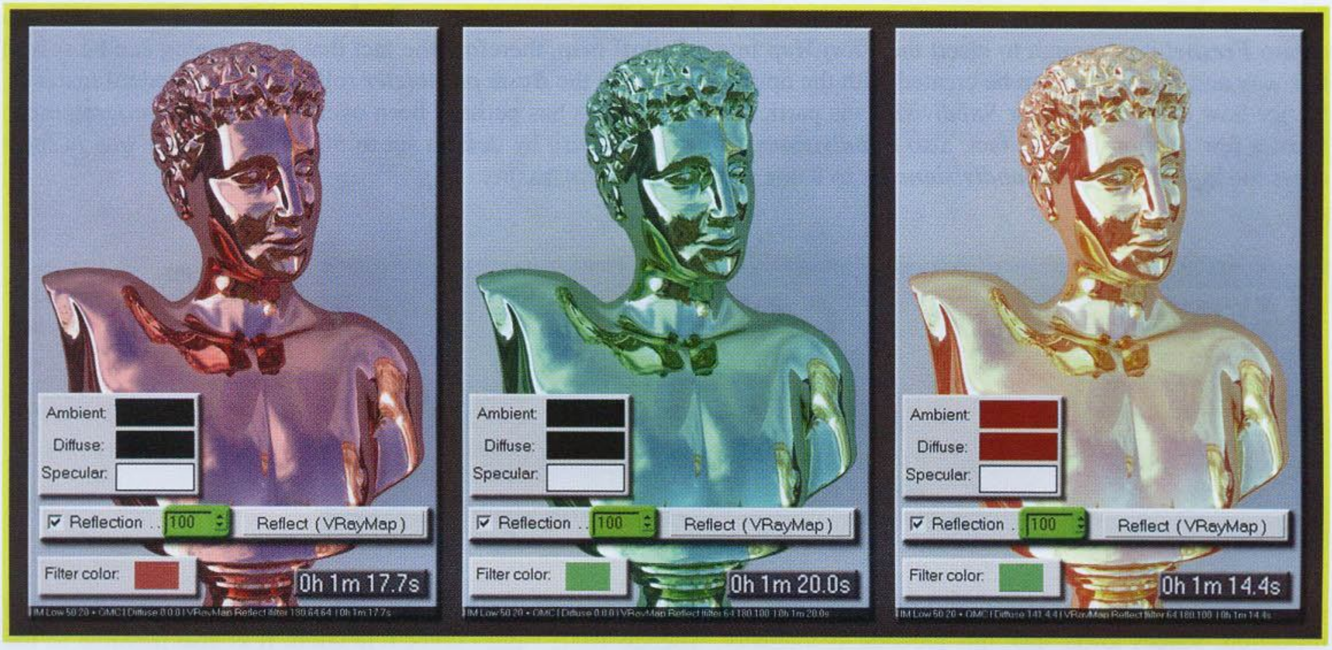
■ **Figure 5.197**
The parameters for the definition of reflections located in the ***VRayMap*** are identical to those in the ***VRayMtl***.

Filter color - this represents the reflection multiplier. Values around RGB = [255, 255, 255] increase the amount of reflection. Values lower than this reduces its intensity, whereas RGB values color reflection. It is also possible to use textures for defining the position and intensity of reflections. It is advisable to leave the ***Reflect channel*** of the Standard material always set to 100 and use ***Filter color*** to lower the intensity of reflections, because the ***Photon Map*** would not detect any change of the spinner.



■ **Figure 5.198**
Different ***Filter color*** values. In the first three examples a Standard material with red ***Diffuse*** color has been used, whereas ***Reflect Filter*** color varies from black to white. The ***Reflect channel*** is always set to 100.
In the remaining tests the only difference concerns ***Diffuse color*** which is now completely black.

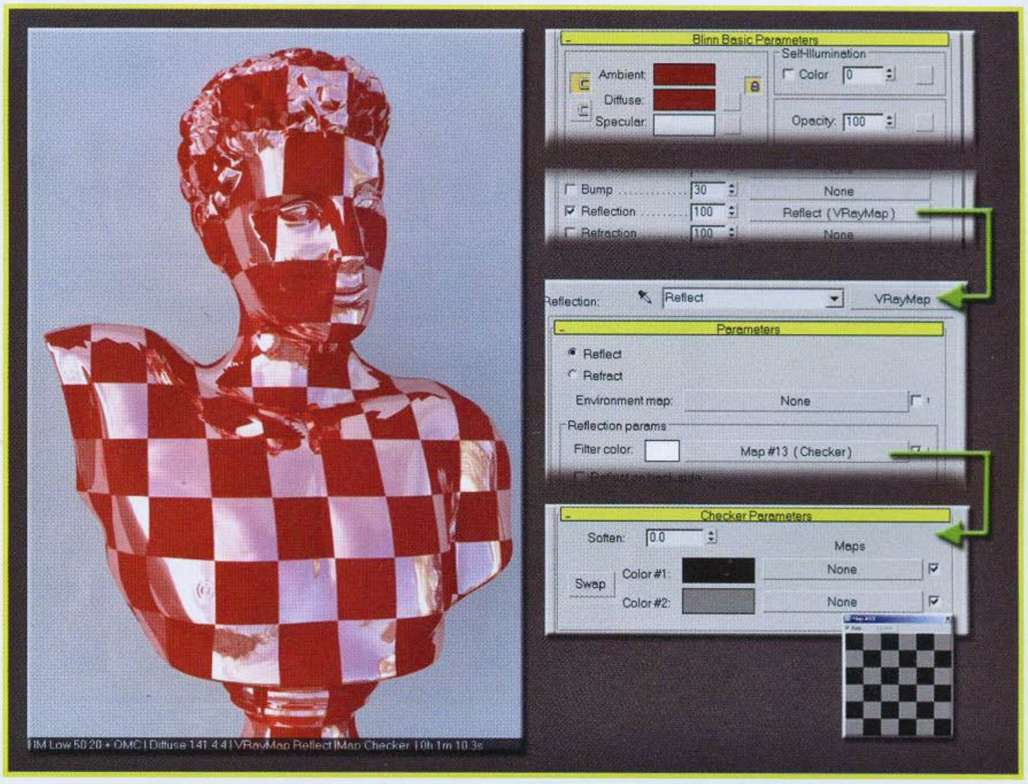
By analyzing the previous test also other things can be noticed. By using a color and a reflection of RGB=[255,255,255] in the ***Diffuse*** channel, the material is not completely reflective. Or rather it reflects but still shows a certain amount of ***Diffuse*** color coming from the Standard material. This is because the 3ds Max shader is not optimized to be efficiently used with ***VRay***. In a similar situation, but with ***VRayMtl*** in the place of the Standard material, the model would be completely reflective and thus obey the law of energy preservation. A completely reflective material cannot have ***Refract*** or ***Diffuse*** attributes. This does not happen with Standard materials, and for this reason it is best to use only ***VRayMtl*** for a physically correct result or in any case use maps and materials installed with ***VRay***.



■ **Figure 5.199**
Different tones of ***Filter color*** being used.

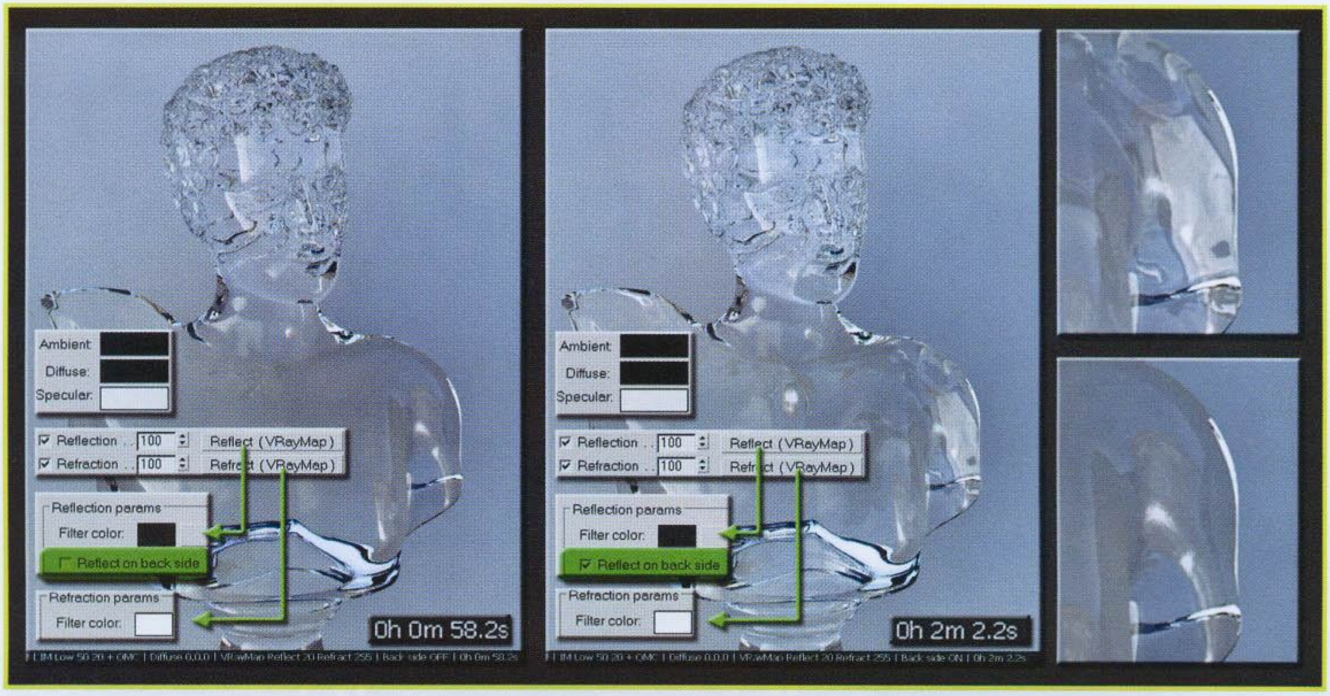
For the ***Filter color*** parameter, as well as simple black and white, one can also assign other colors. When doing so, the color of reflection influences the final result by addition.
In order to employ a map, it is enough to select it with the Browser and insert it in the ***Reflection*** channel. In our case a Checker map has been used. Now ***Filter color*** is no longer taken into account and it is the texture which regulates the intensity, color and position of reflections.

■ **Figure 5.200**
A map being used for the ***Reflection*** channel.



The Checker map, being made of grey tones, ensures that the red ***Diffuse*** color is not altered. The completely black areas of the map make the material completely opaque, instead the grey areas of the texture increase reflection.

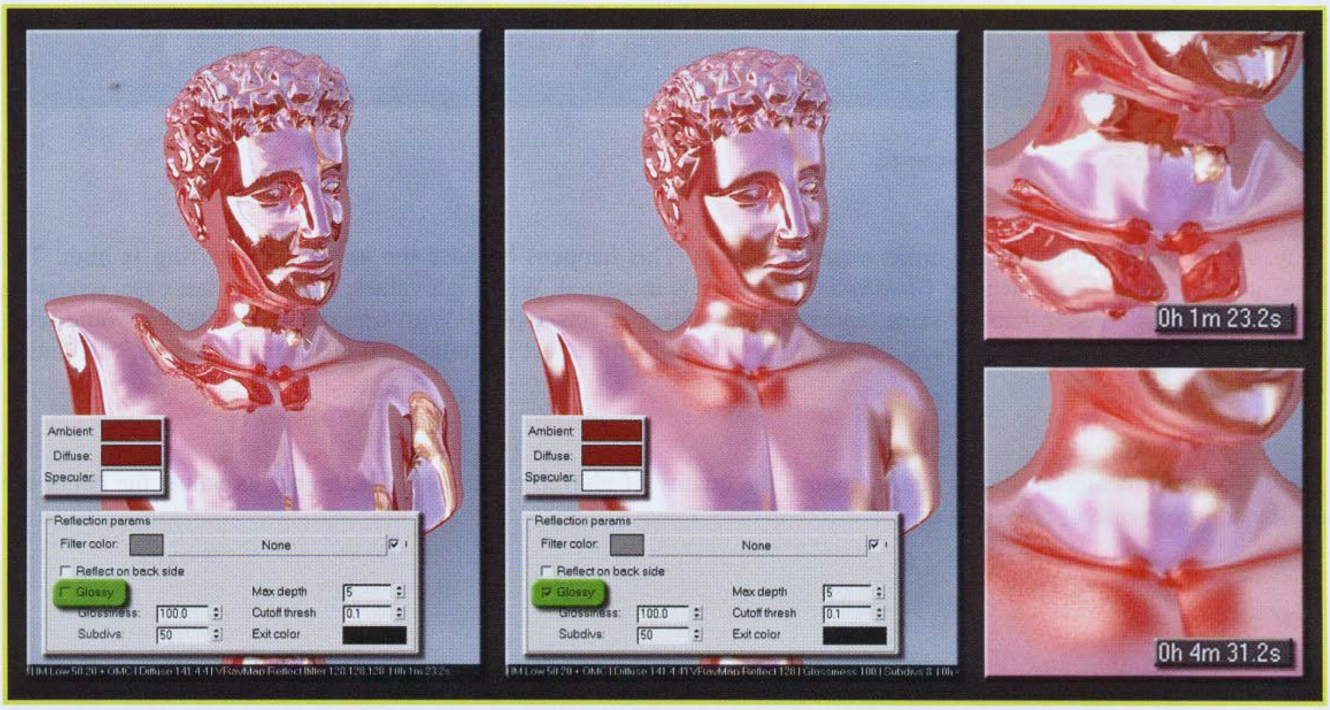
Reflect on back side - this parameter forces the calculation of reflections also on the back side of faces within geometries and generates a better render. By using this parameter, rendering time may increase considerably.



■ **Figure 5.201**
Reflect on back side parameter being used.

In this example, the statue is 100% transparent with a slight degree of reflection. By activating the ***Reflect on back side*** parameter, as well as a considerable increase in rendering time, reflections are calculated also on the inner faces of the statue, improving realism and general quality at the expense of a longer rendering time.

Glossy - this parameter activates blurry reflections.



■ **Figure 5.202**
Activating the ***glossy*** option allows one to create materials with blurry reflections.

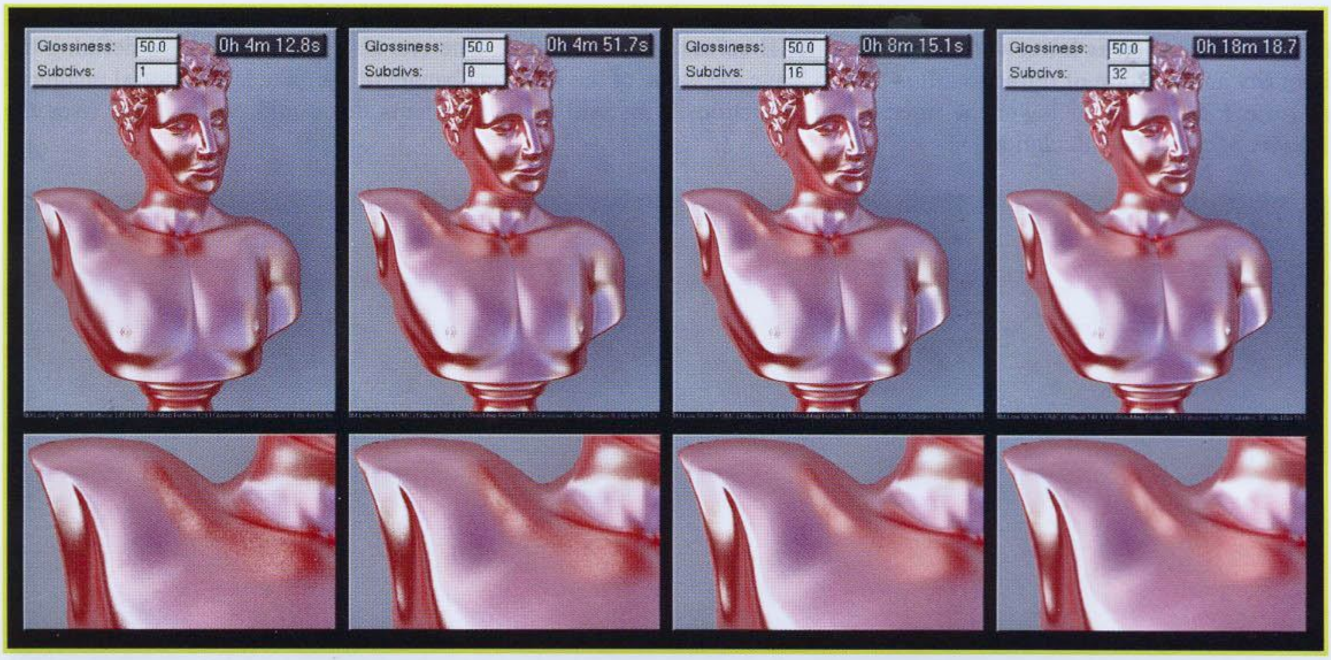
By activating ***glossy*** and using default ***Glossiness*** and ***Subdivs*** settings, rendering time doubles. Compared to ***VRayMtl*** there is a substantial difference: one cannot create ***glossy*** effects via interpolation.

Glossiness - with this parameter one can define the intensity of reflection blurring. Settings close to 0 produce extremely blurry reflections. Settings above 0 generate reflections which are more sharp and precise.

■ **Figure 5.203**
The blur effect decreases as ***Glossiness*** is increased.

With settings near 0.0 rendering times increase greatly. This happens because the amount of blurring is high. The opposite situation occurs when the ***Glossiness*** parameter is increased. With high values, like in the last case, rendering time is in any case higher than the same render without ***glossy*** effects applied. With ***Glossiness*** = 10,000 it would seem like there is no blurring. In actual fact, although it is very little, the effect is applied and this is the reason for calculation time being longer.

Subdivs - this parameter allows one to control the quality of blurred reflections. The higher its value, the better the quality at the expense of rendering time.



■ **Figure 5.204**
L'attivazione del ***glossy*** permette la realizzazione di materiali dalle riflessioni sfocate. - - - DVD - - -

As one expects, by increasing the number of subdivisions, rendering quality increases and therefore also rendering time. In this test, with ***Subdivs*** between 8 and 16, one reaches a high level of quality.

Max depth - this represents the maximum number of times that the geometry can be reflected. When this limit is reached, reflection color is going to be the one to be defined by ***Exit color***.

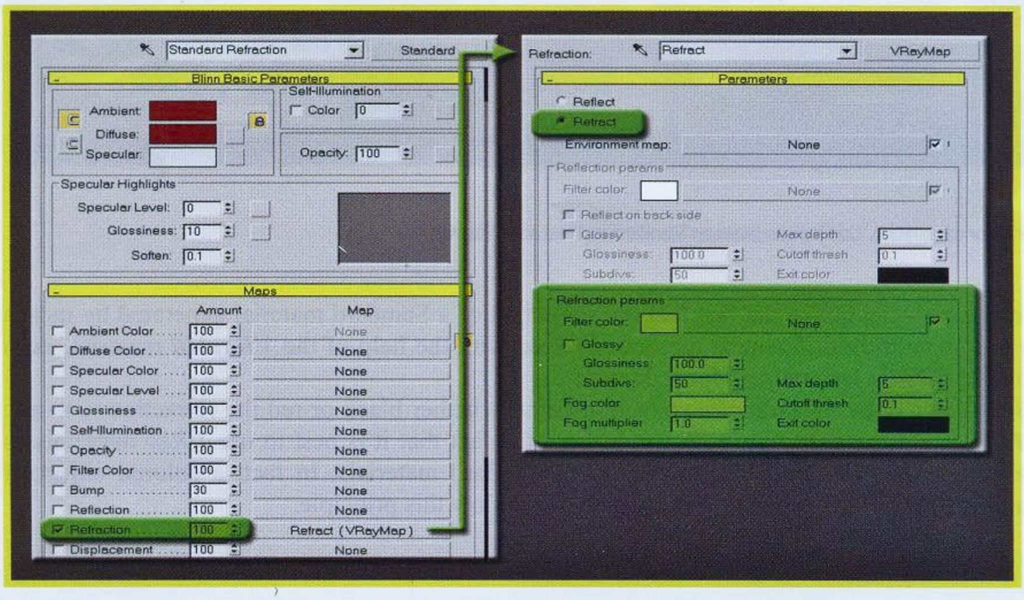
Cutoff thresh - this represents the threshold beneath which reflections are not calculated.

Exit color - this is the color which is given back when the number of reflections defined by the ***Max depth*** parameter is reached.

For these last three parameters, it is advisable to read the chapter concerning the ***VRayMtl***.

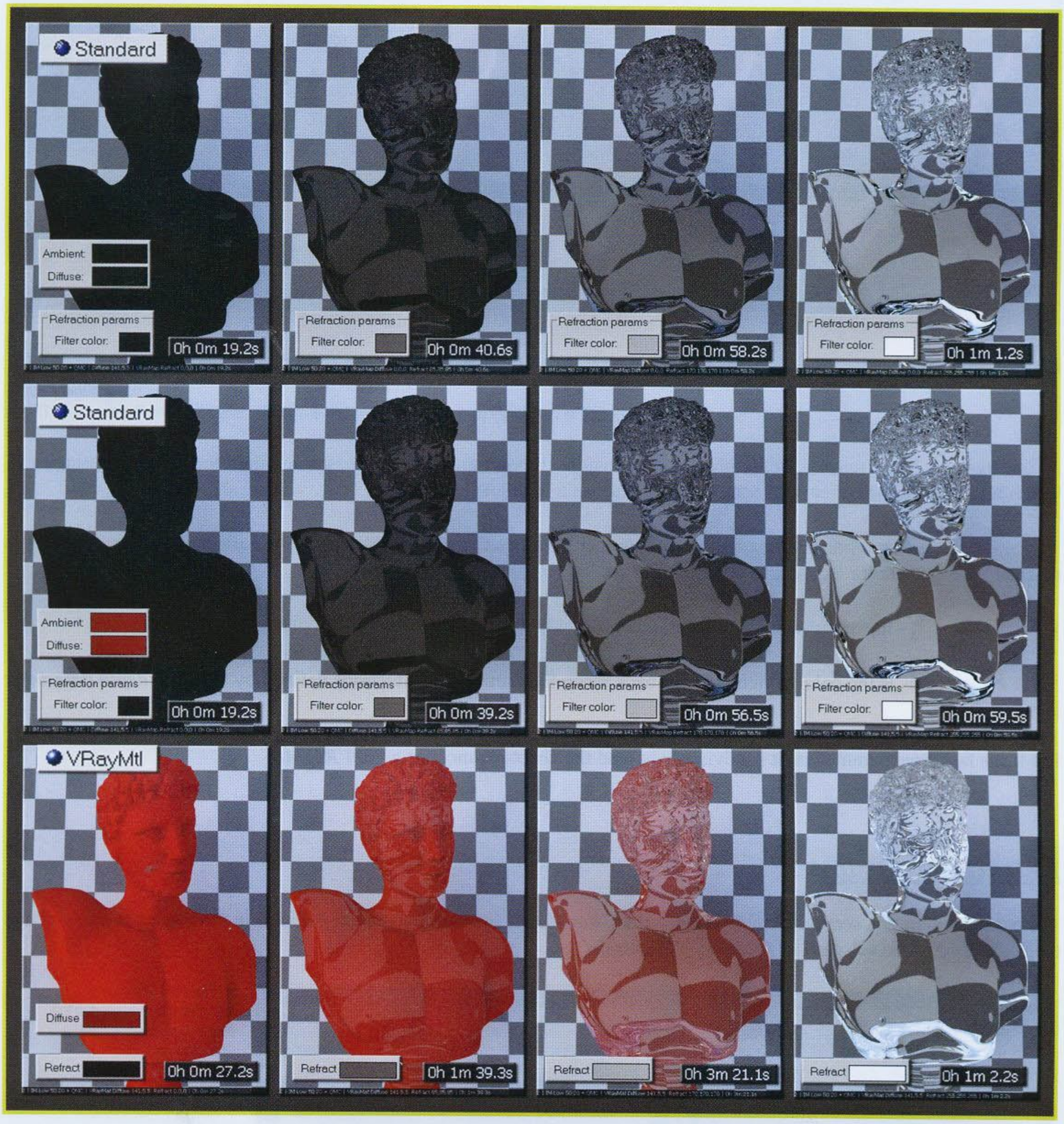
Refraction params

The parameters in this section are active when the ***VRayMap*** is inserted into the ***Refraction*** in the Standard material and the ***Refract*** option is selected by means of the selector



■ **Figure 5.205**
The ***VRayMap*** applied in the ***Refraction channel*** is of a Standard material.

Filter color - this represents the multiple of refraction. Settings around RGB = [255, 255, 255] increase the amount of refraction, whereas lower values reduce it. RGB values color refraction. One can also use a texture for defining the intensity and position of refraction. It is best to leave the value of the ***Refract*** channel of the Standard material unaltered and use ***Filter color*** to lower the intensity of reflections. This will avoid problems that could occur if one uses the ***Photon Map*** for GI calculation.



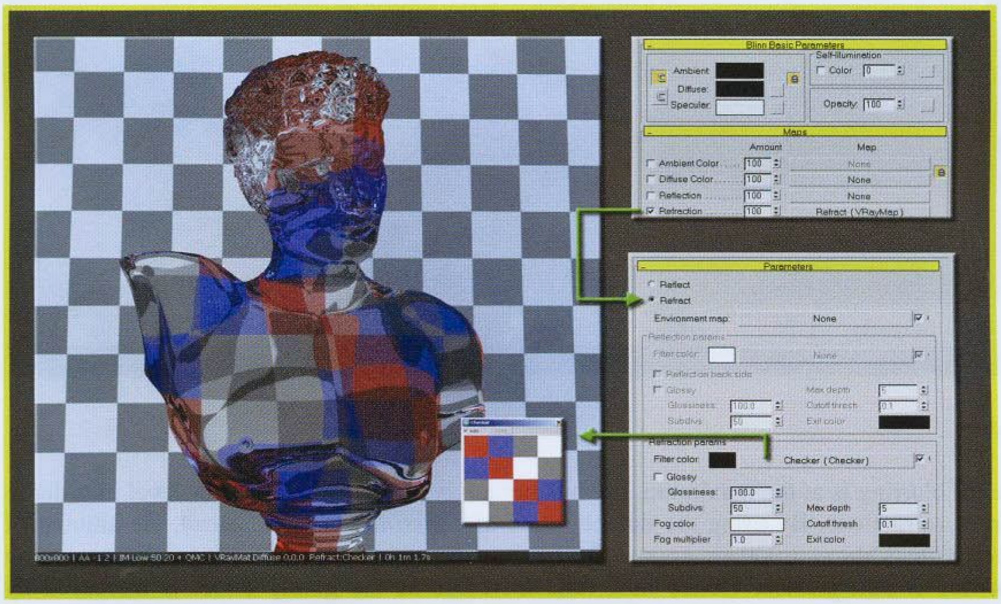
■ **Figure 5.206**
Test with different ***Diffuse*** and ***Filter color*** parameters. Comparison between Standard material and ***VRayMtl***.

By analyzing the previous image, one can notice the fact that the refraction of a Standard material generated by means of the use of the ***VRayMap*** is significantly different to the one obtained with the help of the ***VRayMtl*** material, even using the same colors both for the ***Diffuse*** channel and for refraction.
For Standard materials, ***Diffuse*** color does not influence refraction color: whether black or red the refraction is in any case neutral. With ***VRayMtl*** this is not the case. We have emphasized the fact that it is best to use ***VRay*** materials as they produce physically correct results. This does not happen with Standard materials. In fact, a colored object, if refraction increases, should not lose any color. Standard materials do not follow this principle.

In order to obtain colored with the use of the ***VRayMap*** instead, one must change the color of the ***Filter color*** parameter.

■ Figure 5.207
Color refractions created by means of the ***VRayMap***.

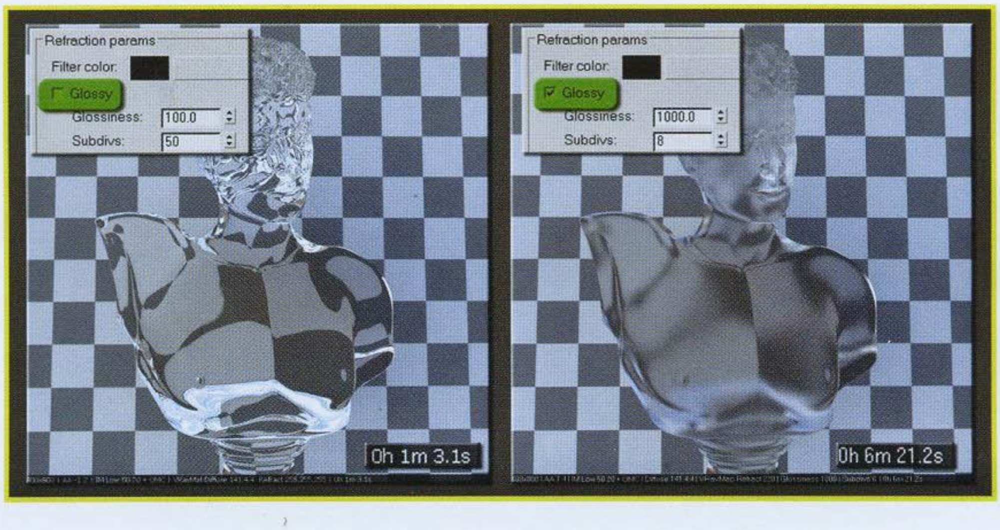
As well as color, one can use a texture to define the position, intensity and color of the refraction.



■ Figure 5.208
A procedural texture being used for the refraction channel. Light grey colors make the material transparent, whereas dark values make it opaque.

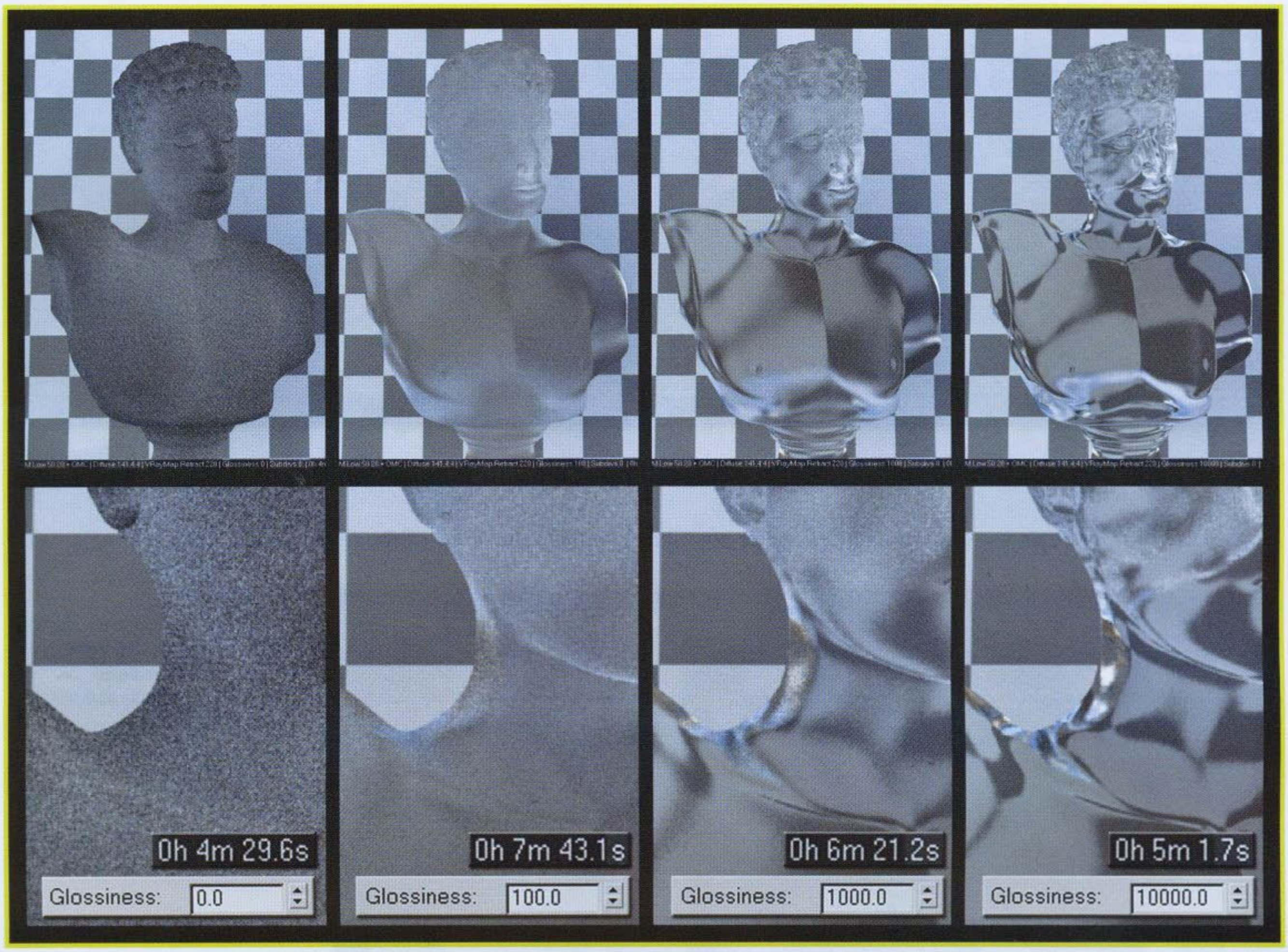
In this case, with a map containing both colors and grey tones, it has been possible to create colored and neutral refractions of different intensities and transparency.

Glossy - by activating this parameter, refractions become blurred.



■ Figure 5.209
This represents the amount of blurring in refractions. Low settings generate very blurred refractions, whereas high values reduce the effect.

Glossiness - this represents the amount of blurring in refractions. Low settings generate very blurred refractions, whereas high values reduce the effect.

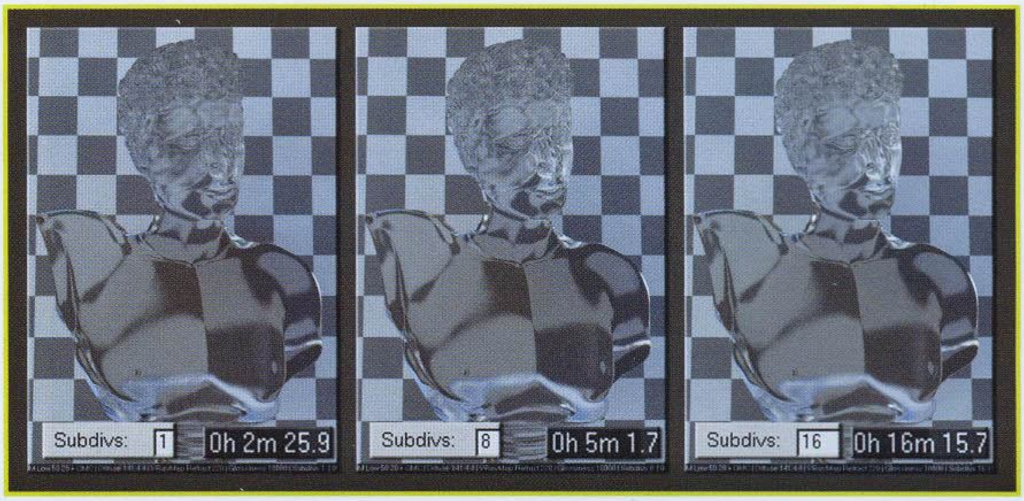


■ **Figure 5.210**
Render with different ***Glossiness*** values.

With ***Glossiness*** = 0.0 the level of blurring is at the maximum, but it is not advisable to use such a low level in order to avoids an incorrect calculation of refractions. By using the default value of ***Glossiness*** = 100 instead, the amount of blurring is very high and correct. Values above 1,000 are enough for a refraction with medium blurring. With very high values such as 10,000, ***Glossiness*** is still present. Notice how time increases as the glossy effect is increased by lowering the ***Glossiness*** parameter.

Subdivs - with this parameter one can decide the quality of blurred refractions. Low values produce renders of scarce quality in a shorter time. High values permit high quality renders but require more time.

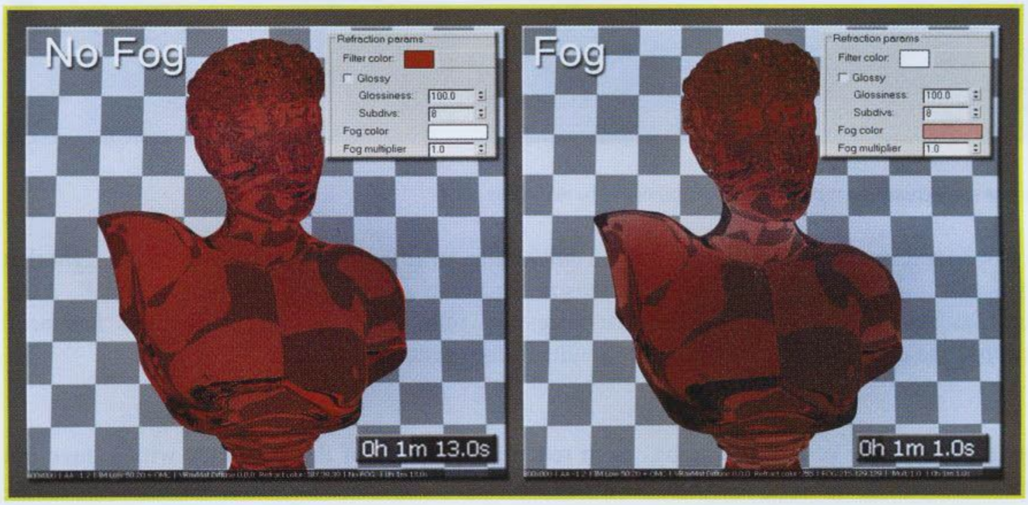
■ **Figure 5.211**
Test carried out with ***Glossiness***=10,000 with different ***Subdivs*** values. As quality and subdivisions increase, so does rendering time.



■ **Figure 5.212**
Image of a detail in the previous test. Notice the difference in quality and noise in the three images. . - - - DVD - - -

In this case geometry is particularly complex and on the face of the statue one can notice flaws due to the low quality of the ***glossy*** effect. A setting such as ***Subdivs*** = 8, which unfortunately is not the default value (it should be IMO), is enough for a good quality render. With 16 subdivisions noise almost disappears, but rendering time increases considerably. With the value 50, the default setting, quality is extremely high but so is rendering time.

Fog color - similarly to what happens for ***VRayMtl***, this value allows one to create materials with characteristics similar to colored glass. This way the thinner parts of models are less colored than the thicker parts.



■ **Figure 5.213**
Fog color being used.

Notice how the thinner zones, such as the neck or the base of the model are brighter and clearer than the thicker parts, like the chest. By using ***Filter color*** only, this does not happen.

Fog multiplier - this represents the multiplier of the ***Fog*** effect.

■ **Figure 5.214**
The intensity of ***Fog color*** is handled by the ***Fog multiplier***.

High values produce very colored and full refractions. Low values produce more transparent and colorless refractions.

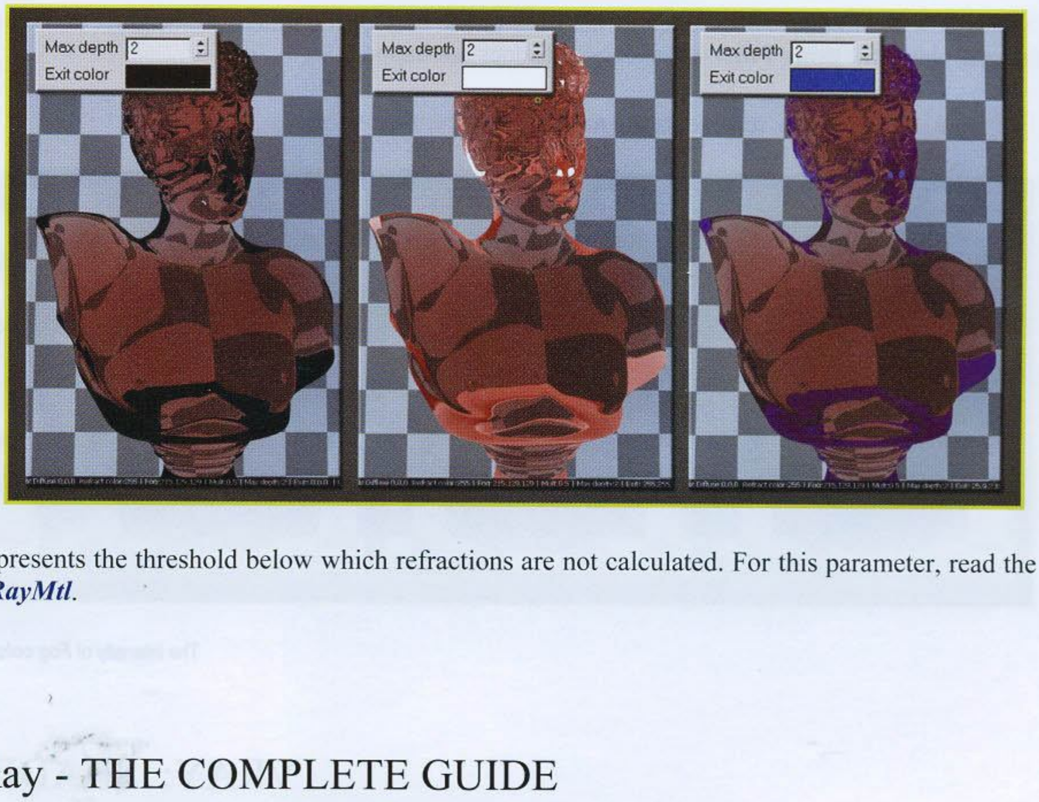
Max depth - this represents the minimum number of refractions that the object produces within itself. Low values make rendering faster at the expense of physically accuracy. High values slow down calculation but produce better renders.

■ **Figure 5.215**
Thanks to the ***Max depth*** parameter one can increase the number of bounces generated by refractions.

With ***Max depth*** set very low, ***VRay*** uses very few bounces for the calculation of refractions and therefore the light that penetrates inside the model is not able to get out. This is the reason for some areas being completely black. Also, rendering time increases as ***Max depth*** is increased. Once the necessary number for correct reflections has been reached, by increasing ***Max depth*** the quality does not change and neither does rendering time.

Exit color - once the maximum number of refractions has been reached, the parts of the geometry which have not been hit by refraction rays are colored with the tone selected in ***Exit color***.

■ **Figure 5.216**
The ***Exit color*** parameter.



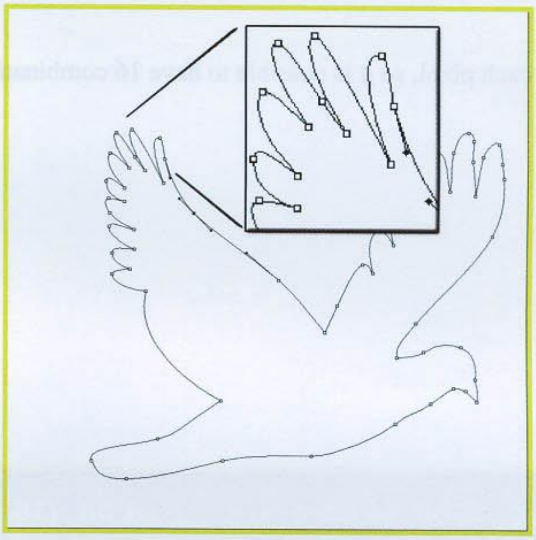
Cutoff thresh - this represents the threshold below which refractions are not calculated. For this parameter, read the chapter concerning the ***VRayMtl***.

VRay Map: VRayHDRI

INTRODUCTION

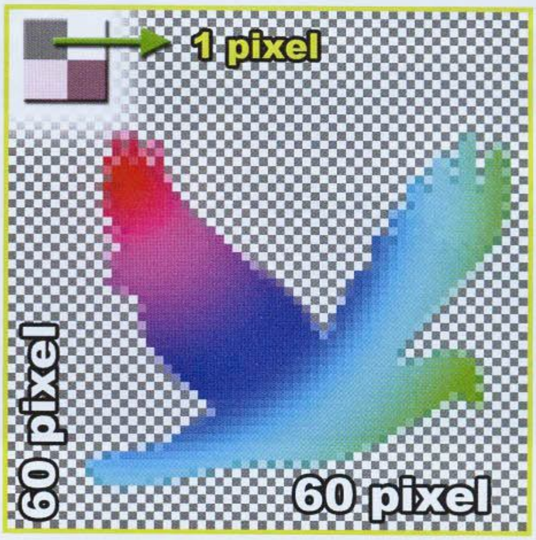
An image generated by a computer can be of two types: vectorial or raster.

Vectorial images are characterized by the fact that the images are represented by means of a group of mathematical primitives such as points, lines, curves and polygons which are accordingly colored. The main advantages of vectorial graphics as opposed to raster graphics is the quality in relation to the loss of information, the greater data compression and the fact that it is easier to manage possible modifications. The defects are the difficulty of fulfilling this type of image, as one must resort to tools such as CAD applications or vectorial graphics and the difficulty of creating images which are rich in color blends.



■ **Figure 5.217**
Vectorial image. Notice how the profile of the drawing is made up of lines and control points which can easily be changed.

Raster images are the most common type and they are typically created via digital cameras, scanners and so on. In this case the PC subdivides the image into very small squares called pixels, which contain information about their color.



■ **Figure 5.218**
Raster image. Notice the jagged edges due to pixelization and the high quality of blending.

In computer graphics most renders are made with raster images.

In the chapter concerning ***Color mapping*** the subject of ***Color depth*** was discussed. What one intends with the term is the amount of bits needed to represent the color of a single pixel in an image. This way one can have images of:

1 bit ..... (2^1= 2 colors).
2 bits ... (2^2= 4 colors).
4 bits ....(2^4= 16 colors).
8 bits ....(2^8 = 256 colors).

For example, in a 1-bit image, each pixel has only 1 bit for determining its color, that is:

[0] = white
[1] = black

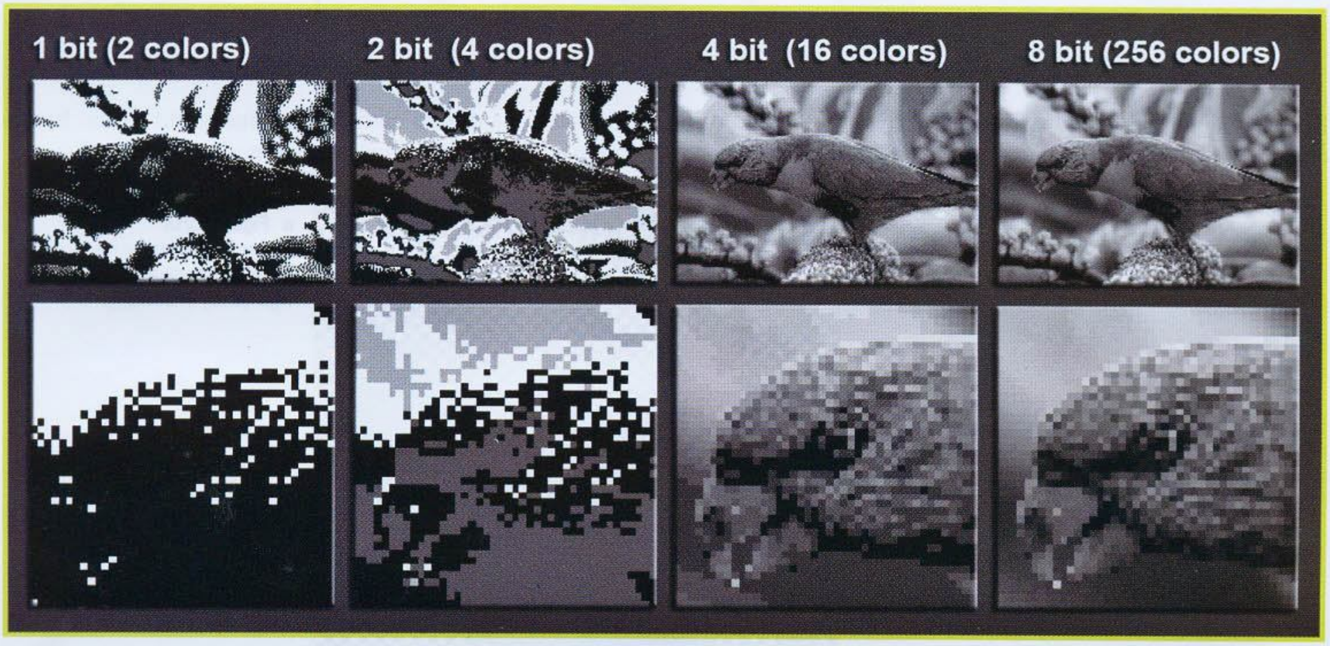
An image at 2 bits has 2 bits for each pixel and therefore one can have 4 color combinations for each pixel.

[0,1] = white
[1,0] = light grey
[1,1] = dark grey
[0,0] = black

A 4-bit image has 4 bits for each pixel, so it is possible to have 16 combinations of different colors for each pixel:

[0,0,0,0] = white
[0,1,1,1] = grey1
[0,0,1,1] = grey2
[0,0,0,1] = grey3
[1,1,1,1] = grey4
[1,1,1,0] = grey5
[1,1,0,0] = grey6
[1,0,0,0] = grey7

etc…



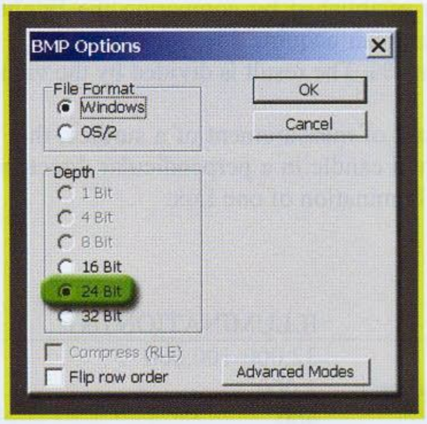
■ **Figure 5.219**
mages with different color depth.

Until now we have discussed only black and white images. The most a pixel can have for a scale of grays is 256 colors. When one involves color images, what happens?

Quite simply, for an 8-bit image, instead of having 256 tones of grey, we would have:

2^8 red colors = 256 variations of the color red.
2^8 green colors = 256 variations of the color green.
2^8 blue colors = 256 variations of the color blue.

If we have 256 tones available for red, 256 for green and 256 for blue, one can obtain $2^8 \times 2^8 \times 2^8 = 2^{24} = 16{,}777{,}216$ colors. In this case one defines this as 24-bit images.



■ **Figure 5.220**
Screenshot of the save window in Photoshop. By saving in .bmp format one can choose color depth. By default it is set to 24 bits.

It can happen sometimes that one finds oneself face to face with a 32-bit image. This does not mean that these images have more colors than 24 bit images, but that there is an extra channel. This is the alpha channel, a channel in a scale of grays which contains information on the transparency of the image. In this case the mathematical operation is equivalent to $2^8 x 2^8 x 2^8 x 2^8 = 2^{32}$.

Before moving on to the concept of ***HDRI*** one must understand the meaning of the word ***floating point*** and ***dynamic range***.

The floating point system is one of the formats by which a number can be written and displayed by digital means. The simpler digital numbers represent a whole number in binary language, with each digit of the number represented by one information bit. This does not permit the memorization of a fractionary number and so other formats have been developed, amongst which the floating point format which is the most widespread. In a floating point representation, a number can be decomposed into two parts, called mantissa **m** and exponent **e**.

Number = mantissa * b $^{\text{exponent}}$

NUMBER	MANTISSA	EXPONENT	FLOAT NUMBER
12345	12345	0	12345×10^0
123450	12345	1	12345×10^1
1234.5	12345	-1	12345×10^{-1}
5.4321	54321	-4	54321×10^{-4}
32451000	32451	3	32451×10^3

The mantissa is a whole number and contains the most significant digits of the number, without taking care of whether the digits are tens, units, decimals and so on. The exponent is another whole number which indicates the relative position of the floating point or the decimal point within the mantissa. As they are whole numbers, the mantissa and exponent can be memorized and elaborated conveniently with the same systems used for whole numbers and this provides a quick system for memorizing and storing fractionary numbers.

The term dynamic range indicates the ratio between the strongest and weakest light which can be captured by the CCD sensor of a digital camera, once the shutter opening time, aperture and ISO sensitivity has been fixed. The dynamic range is therefore a synonym for maximum contrast between light and dark, correctly carried out by the CCD. Also the human eye and classic film have a limited dynamic range. The fact is evident when one thinks of the effect of a torch shone in ones face or when one takes a photograph against the light. Basically, a dynamic range which is too small leads to images which have parts which are too bright and therefore overexpose, or too dark and underexposed. In these areas details are completely lost.

The dynamic range is not usually declared by producers of digital or traditional cameras, but it can be measured by means of certain tests. The measurement is expressed in ev (equivalent value) or in contrast ratio (e. g. 400:1). Moreover, this range varies as ISO sensitivity settings vary, and has its maximum value in correspondence of low ISO, for example 100, and quickly draws lower with higher ISO values. With ISO at 800, dynamic is on average 4 times as low as when ISO is set to 100. In most devices, dynamic range for ISO 100 ranges from 100:1 to 400:1, according to the model.

In order to better understand how dynamic range is measured, we will use the example of PC monitors. The same concept can be applied to cameras, but with the opposite concept. Whereas monitors reproduce images, cameras acquire them. The contrast ratio is established by comparing the brightest white with the darkest black that the screen can reproduce or that the camera can acquire. More specifically, it is the difference in intensity between a 100% white signal and the same signal at 0%. The result is divided by the value of a 0% white calculated in Lux.

Lux (lx) is a photometric unit of measurement of a surface, that is, of the flow of light flow which hits each unit of a surface. The light flow from a candle in a perpendicular direction and at the distance of one meter on a surface of one square meter produces the illumination of one Lux.

1 lx = 1 cd/mq

LIGHT SOURCE	ILLUMINATION (lx)
Sun	32,000-100,000
Reflectors for television studios	1,000
Office lamp	400
Moon	1
Star	0.001

In order to mathematically establish the level of contrast, one must divide the intensity of This mean white by the intensity of black. This means that for a screen, which has for its maximum and minimum are correspondingly 200.5 cd/mq. and 0.5 cd/mq in standard models, the contrast ratio is $(200.5 - 0.5)/0.5 = 400.1$.

The dynamic range in nature is very high and can reach a contrast of 100,000:1. An ***HDRI*** (**H**igh **D**ynamic **R**ange **I**mage) stores the entire dynamic range of light for each of its pixels. As a result, an ***HDRI*** must be encoded in a particular format which allows one to store a vast quantity of data, for example by means of a floating point system and more precisely one with 32 bits per channel, which gives us an image of 96 bits. Another feature of ***HDRIs*** is the fact that they store data with a linear system. This means that the value of pixels is proportional to the amount of light measured by the camera. In this sense, it is possible to define an ***HDRI*** as scene-referred, in other words which represents the original values captured in the moment the photo is taken.

An image, in order to be considered ***HDRI*** and not ***LDRI*** (**L**ow **D**ynamic **R**ange **I**mage) must have certain characteristics. The main difference is given by the number of bits per channel. It is also true that the number of bits alone can be enough to define whether or not an image is ***HDRI***. In fact by converting a ***.jpg*** in an ***HDRI*** with Photoshop, the number of bits per channel increases. However, by doing this one does not create a real ***HDRI***.
The following ***HDRI*** and ***LDRI*** image formats are the most commonly used in CG:

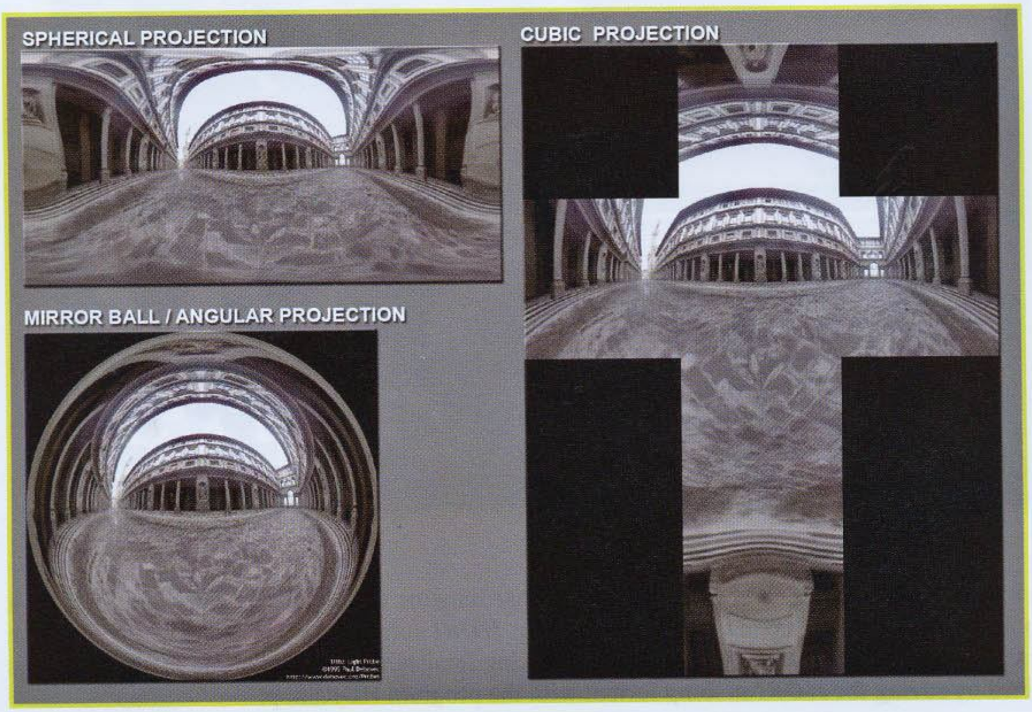
.8 bits per channel : (24 bit image). Classic ***.jpg*** or ***.bmp*** formats. They are to be considered ***LDRIs***.
.32 bits per channel : (96 bit image). Images in ***HDR*** or ***EXR*** should be considered ***HDRI*** images. Unlike formats at 8 and 16 bits, which can store a limited number of values, 32 bit images are encoded with floating point numbers, so they can store an unlimited number of values. It is important to take note that storing information in ***HDR*** format at 32-bit is a necessary condition for an ***HDRI*** but is not enough. If the original image has not captured the whole dynamic range of the scene, it will remain a ***LDRI*** no matter what method has been used to store the data. As the human eye has a dynamic range of 10,000:1, an ***HDRI*** has a clear advantage over an ***LDRI***.

Most digital cameras can acquire a limited dynamic range, which is determined by exposure settings of the camera. This explains why ***HDRI*** images are commonly generated by photos of the same scene made with different levels of exposure of the camera. Once images have been acquired with the use of one of the many commercial software products available, such as HDRshop or Photoshop CS2, they can be "put together" thus creating an ***HDRI*** image.

■ **Figure 5.221**
Jpeg photos taken with a digital camera at different exposure levels, composited in HDRShop in HDRI format and retouched in Photoshop.

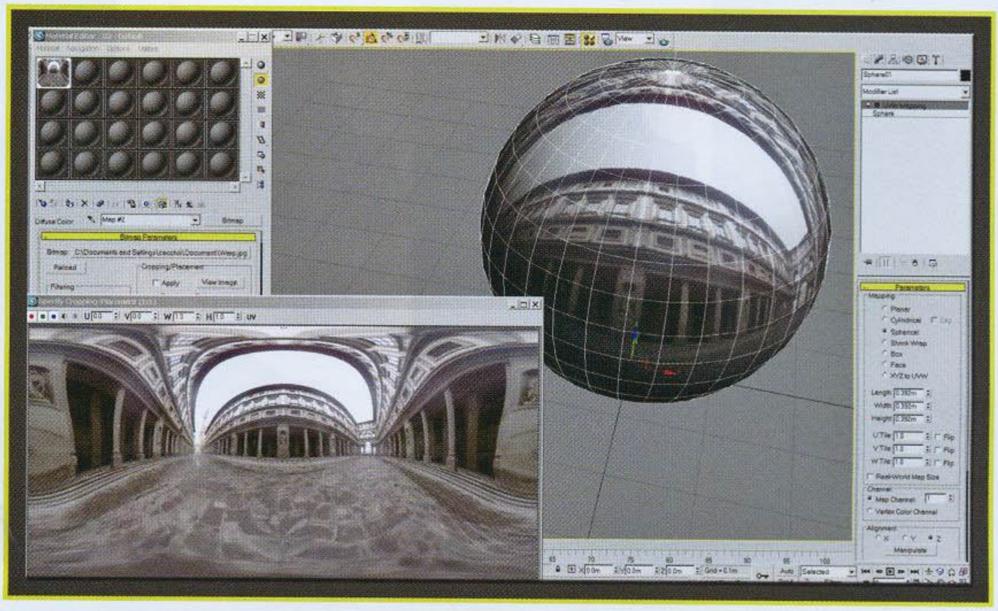
In the previous image a series of photographs, taken with different levels of exposure, has been imported into HDRShop and united so as to form an ***HDRI***. After it was imported into Photoshop CS2 for a slight correction of exposure and range.

Now the question is what use can be made of this type of image within *VRay*. ***HDRI*** images are used for illuminating a 3D scene realistically with a technique called ***Image Based Lighting***. By taking one of these maps, which contain the entire luminous range of the "real world", the Global Illumination algorithm of *VRay* uses this information for "generating" light by using the information of single pixels of the ***HDRI*** map.
There is however a problem. In order to calculate the GI via the ***HDRI*** map, the latter must represent the surrounding environment for all 360°. If we take a look at the previous ***HDRI***, one can see that it is in fact a small portion of a 360° space which was surrounding the photographer when the picture was taken. For this reason ***HDRI*** maps, in most cases, are produced in a particular format:



■ **Figure 5.222**
The three main formats which one usually uses to produce ***HDRIs***. Thanks to the way they are made, these images can be mapped on a 3d sphere so that one can dress it without any problems occurring.

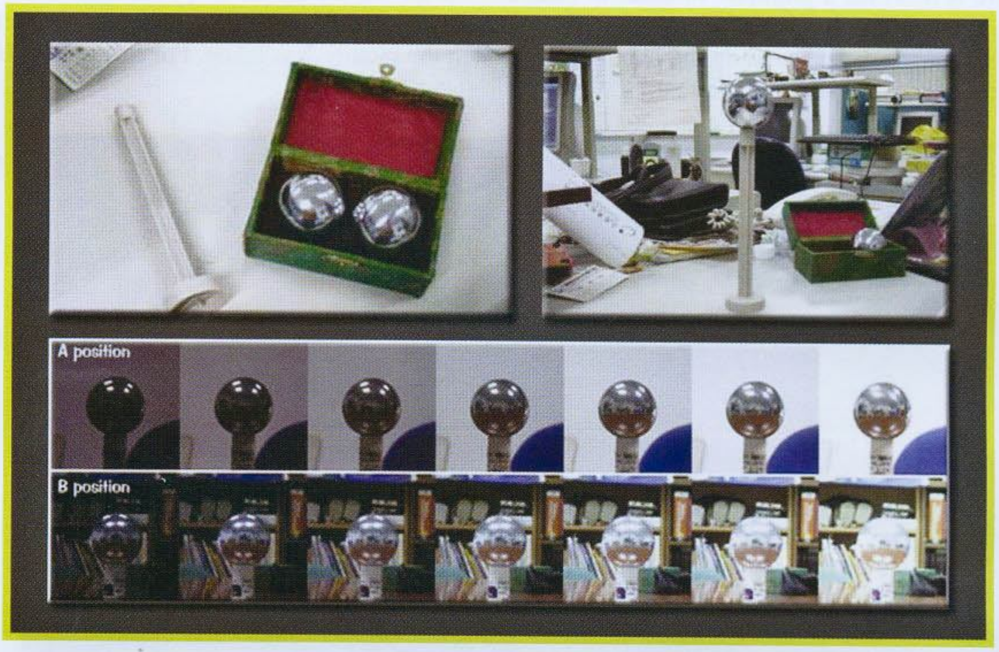
These three maps can be used for texturizing a sphere so that it can be covered without any flaws.



■ **Figure 5.223**
Texturization of a spherical primitive with a spherical ***HDRI*** texture.

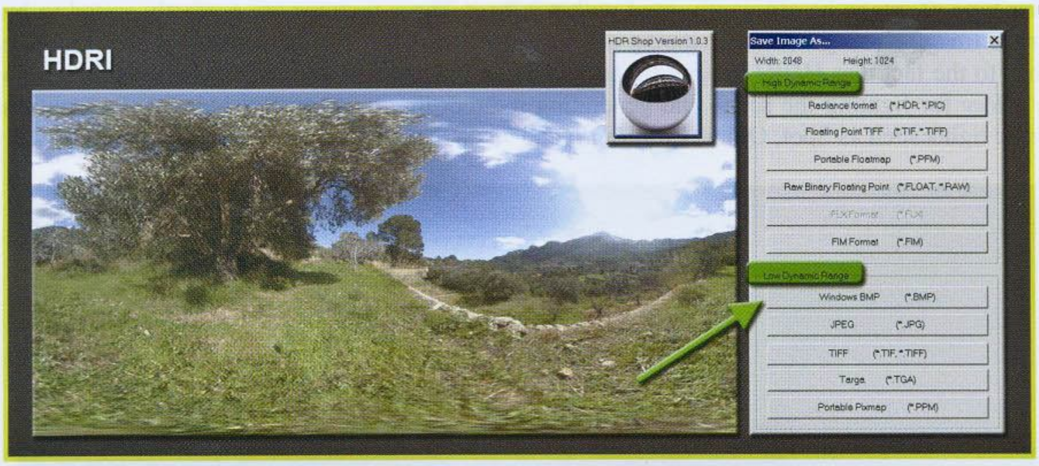
From the technical point of view, all three formats produce the same type of illumination. However some 3D programs favor one format in particular.

The creation of a 360° HDRI is a very simple operation and anybody can do it at home. It is enough to have a tripod, a digital camera and a perfectly smooth and chromed sphere. The larger the radius of the sphere is, the higher the quality of the image, especially because the camera can be placed at a greater distance from the sphere to be photographed and therefore it influences the image less with its presence. One can erase the camera from the mirror ball if one wishes, by means of certain superimposition techniques.



■ **Figure 5.224**
The main steps for creating a do-it-yourself ***HDRI*** map

Once one has obtained the images at different levels of exposure, all that remains to be done is elaborate them in a program such as HDRShop and then finish by saving the final image in ***HDRI*** format. At this point we have a true ***HDRI*** image. In order to grasp the meaning of ***HDRI***, observe the following images. The first is a true ***HDRI***, whereas the second is an ***LDRI*** obtained by saving the ***HDRI*** in .bmp format with the program HDRShop.



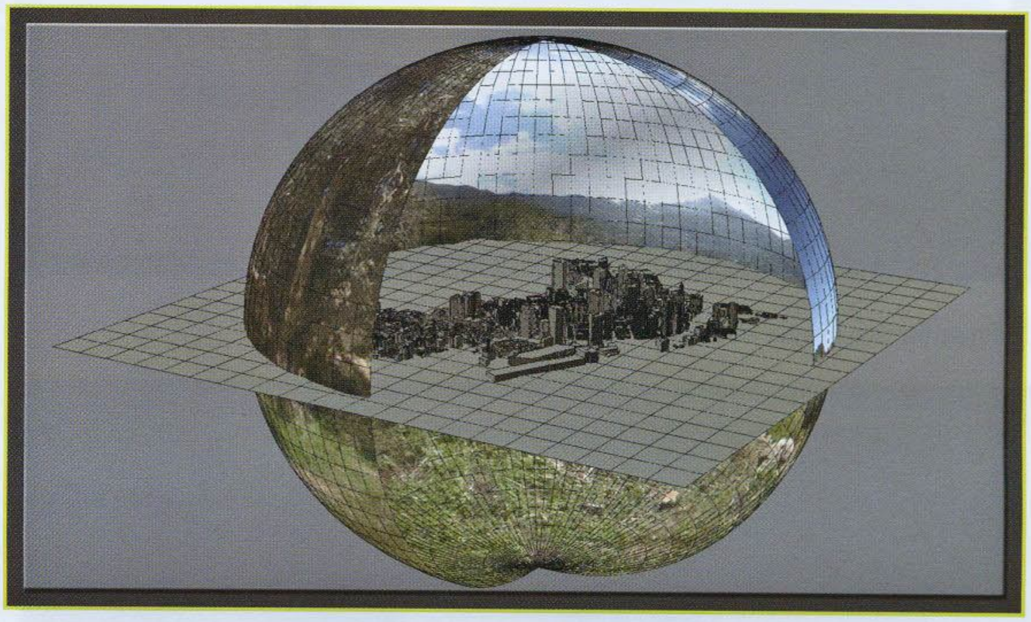
■ **Figure 5.225**
HDRIShop is a freeware program for the elaboration of ***HDRI*** images. To download it visit the following site: http://www.hdrshop.com/
We must mention the creator of this program, ***Paul Debevec***, the pioneer of ***HDRI*** images.

Compare the two images at different exposure levels

■ **Figure 5.226**
HDRI and ***BMP*** images with different exposure levels compared. The "starting images", that is, the ***HDRI*** and its ***LDRI*** conversion, appear identical. By lowering the exposure time, in the ***HDRI*** image luminous details are perfectly visible at the height of the sun and in the areas with most light concentration. This is because the monitor sees the pixels as completely white. In actual fact they are not, they contain much more information than the monitor is able to read. So by lowering the exposure, this data comes "into the light". With .bmp or in general ***LDRI*** images this is not possible, as pure white, which is RGB=[255,255, 255] is either white or not because the monitor shows exactly what the pixel represents. By lowering the exposure, all one does is modify luminosity, as would happen in a photo re-touching program. By increasing exposure, the two images appear identical because the monitor is not able to reproduce such high luminosity values, but in actual fact the ***HDRI*** contains much more information.

The main asset of ***HDRI***s, especially for the 360° panoramic type, as they are generated by a series of images taken in the real world, they have the trait of reproducing a space which really exists.

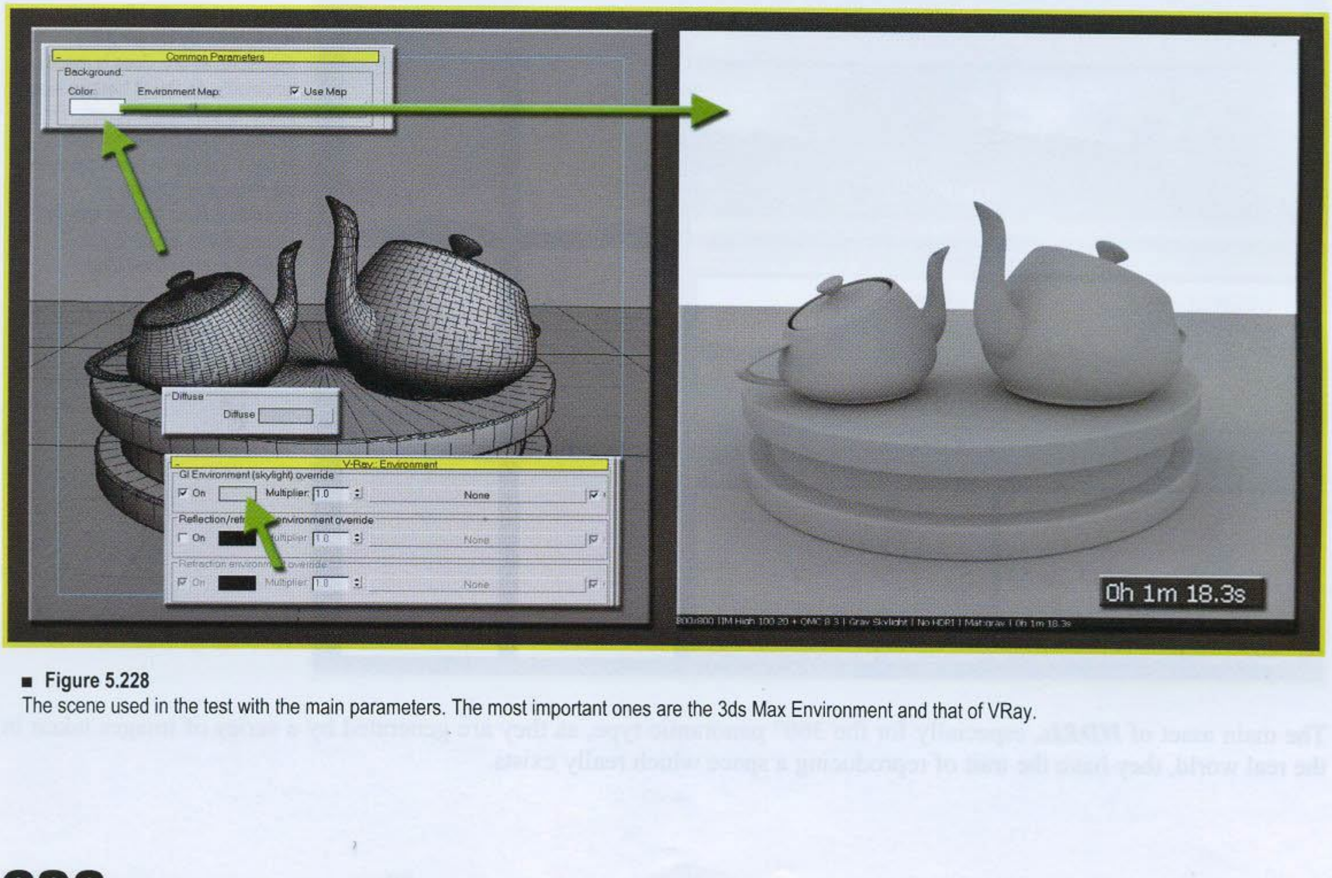
But how can such a peculiar type of image be efficiently used within *VRay*? These textures are perfectly suited to be mapped on sphere geometries. Imagine using one for mapping a huge sphere with an infinite radius, within which the 3D scene is to be set. Thanks to the fact that ***HDRI*** images contain so much information and their dynamic range is so wide, they can be used by *VRay* for lighting the scene and generating indirect illumination. But indirect illumination only, as ***HDRIs***, unlike classic lights such as Spot, Omni, ***VRayLight*** and so on, cannot generate direct light.



■ **Figure 5.227**
Schematic representation of an ***HDRI*** image within a 3D program. In actual fact VRay does not create a geometrical sphere. The ***HDRI*** image is mapped directly in the Environment of 3ds Max or VRay in an invisible way, as we will see shortly. All the user need do is specify the type of map, its orientation and intensity.

PARAMETERS

After this long introduction, we must put into practice what we have learnt so far. First of all we will analyze the scene which is to be used in the next few exercises. It is made up of four geometries, to which a simple grey material has been added. This is illuminated by a grey ***skylight***, generated by the ***VRay Environment*** and calculated in GI with the ***IM+QMC*** system. The background instead, is managed by the 3ds Max Environment, which has been set to white for this occasion.

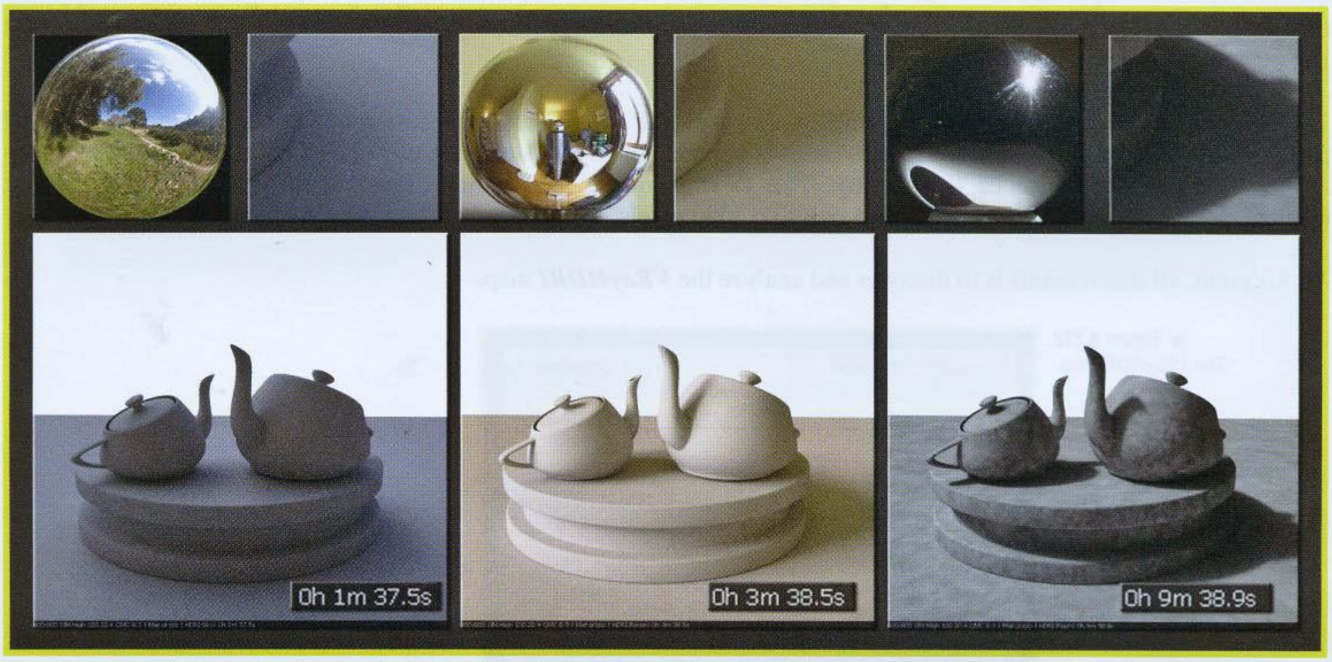


■ **Figure 5.228**
The scene used in the test with the main parameters. The most important ones are the 3ds Max Environment and that of VRay.

In the previous case ***HDRI*** has not been used as ***skylight*** generates the illumination needed. We will now use an ***HDRI*** in the place of simple grey ***skylight***.

■ **Figure 5.229**
The ***HDRI*** map has more depth, color and directionality compared to simple ***skylight*** color.

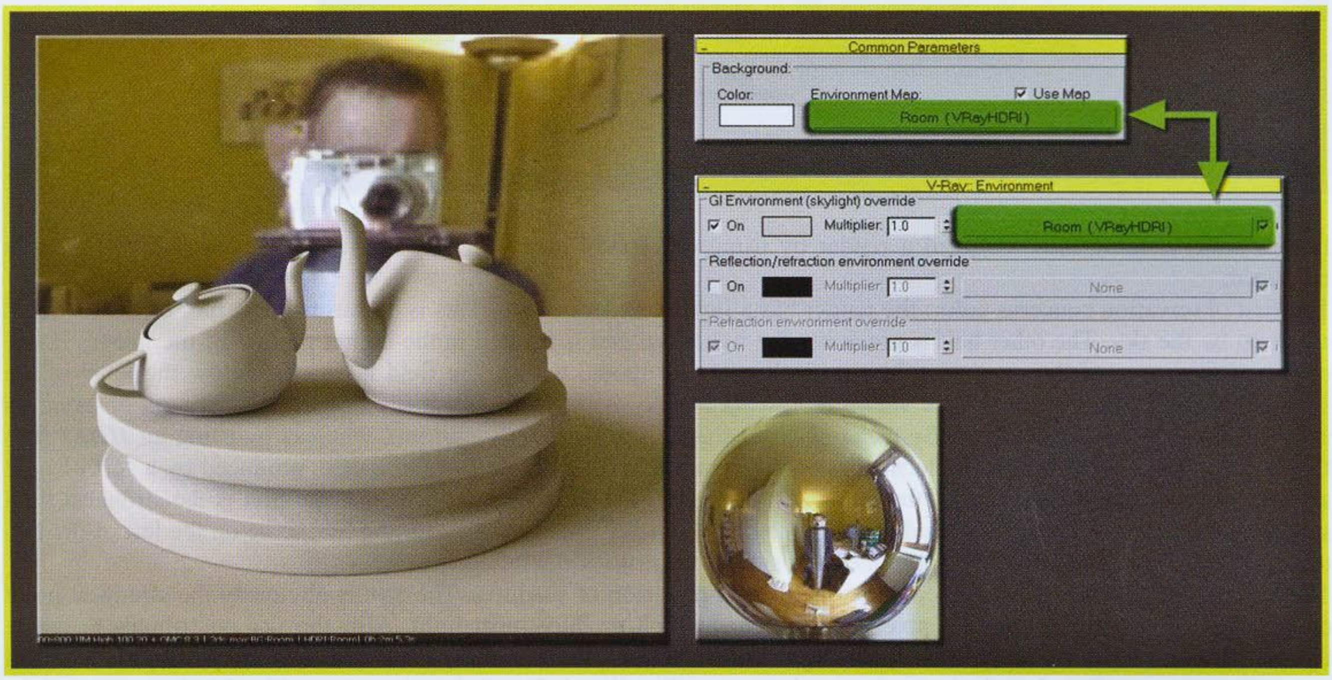
Before understanding how to fulfill a render by using the ***HDRI*** map, we will analyze the render which has been produced.
The first observation concerns color: the render has a light blue dominance. This is because almost half of the ***HDRI*** is taken up by the sky. The other half is green but the light blue part is located in the higher portion of the image. If one observes the image of the mapped sphere with the 3D city model within, the field is located below the level of the streets and the portion of the sphere which represents the sky is mapped with a sky texture. The second observation concerns the position of shadows. By using ***skylight*** shadows are perfectly uniform, whereas by using an ***HDRI*** map they are emphasized. In this example shadows are generated in the right part of the image, produced by a sort of light source coming from the left side of the image. By analyzing the ***HDRI*** one notices that the Sun is the strongest light source. In the test, the ***HDRI*** is placed on the "imaginary sphere" in such a way for the Sun to be on the left of the two teapots. This allows shadows to form on the right side of the teapots and basements.
The third and last observation concerns the quality and precision of shadows. The Sun is obviously the strongest point of illumination in the map and its pixels are the ones which produce most light. Also the sky contributes to create illumination, but it is a more uniform sort of light which cannot create defined shadows. The scene portrays a beautiful day with an intense Sun, but shadows are not sharp or well-outlined. This is because a map like this is not enough to produce well-defined shadows, although it is an ***HDRI*** with high resolution and quality. The only way to obtain sharp shadows is to use an ***HDRI*** with a high level of contrast. For this purpose we would need maps with very few but extremely luminous pixels and the remaining ones completely black.



■ **Figure 5.230**
Different ***HDRI*** maps being used. A big thanks to flipside for providing the ***HDRIs***.

In the previous image various ***HDRIs*** were used. The peculiarity which distinguishes them is the contrast between the zones in the shade and the well-lit ones, as well as the intensity of the light source itself. In the first case both Sun and sky contribute towards the illumination of the scene. In this case shadows are very gentle. In the second case the shadows are produced by the window which, though illuminated by the Sun, is in strong contrast with the rest of the space which is made up of the walls of a bedroom. In the last example the image is almost completely dark, except for the small dot of light belonging to the flash of the camera. This allows one to obtain very defined and well-outlined shadows. It is however obvious that using high contrast ***HDRIs*** leads to another problem. If the ***HDRI*** is able to reproduce sharp shadows, many samples will be necessary to obtain a good render quality and rendering time will therefore be consistent.

In order to analyze the effect of the ***HDRI*** map better, the same image will be used as a background image, so it is necessary to insert it into the 3ds Max Background. This means that both GI and background are managed by the same map.



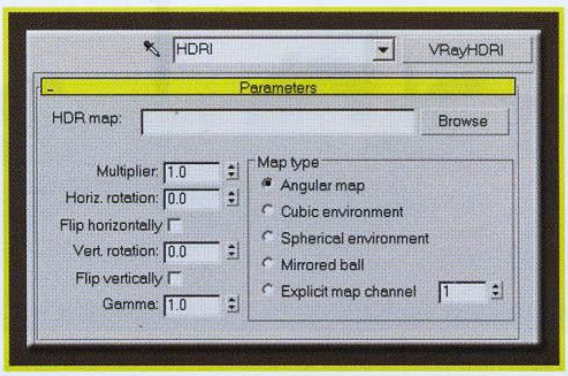
■ **Figure 5.231**
The ***HDRI*** being used as a background image.

If we take a look at the render, we can make a few points. The first is that the photographer, who is being reflected by the sphere, has the light source on his right. This means that he could see shadows going from right to left, the opposite of what happens to the 3D model in the scene. One can state that by using an ***HDRI***, with default parameters, the object is rendered specularly compared to reality.

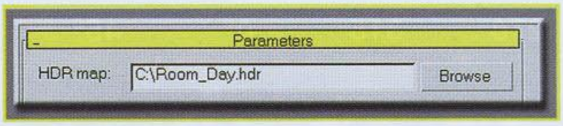
The second point concerns the scarce ***HDRI*** quality with the consequent pixelization of the background. The ***HDRI*** measures 1024 x 1024 pixels. A resolution which is not that small. If instead one uses it as a background image to map a 360° space, it appears to be low quality. This is why lots of sites which offer and sell ***HDRI*** images suggest resolutions which are extremely high, such as 8000x3000 pixels.

At this point, all that remains is to discover and analyze the ***VRayHDRI*** map.

■ **Figure 5.232**
The ***VRayHDRI*** map.

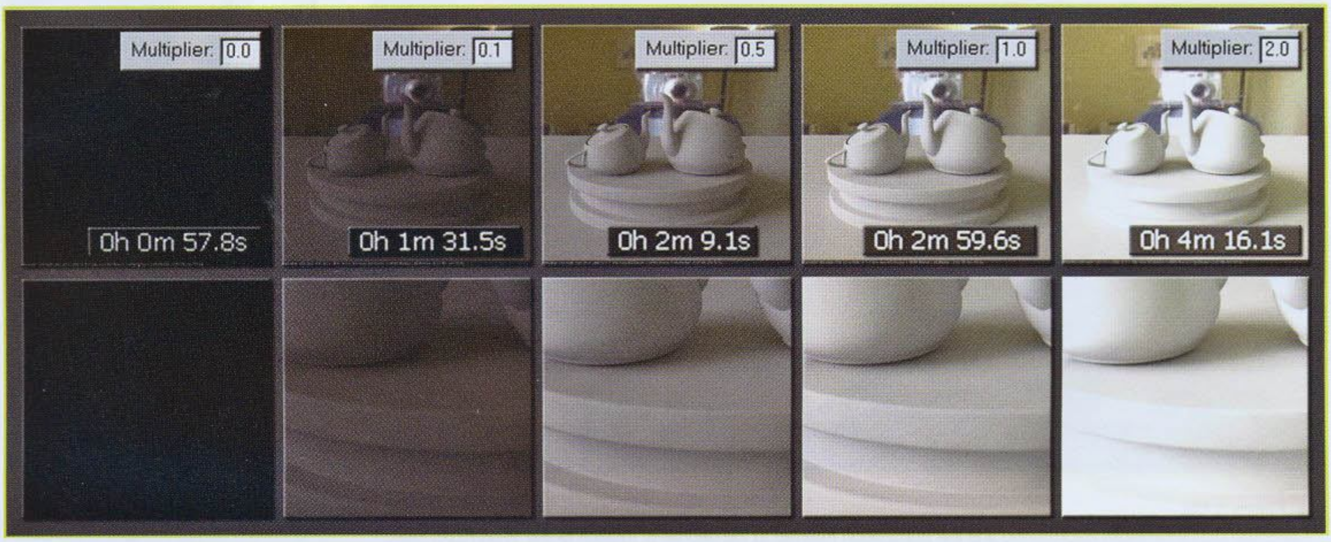


HDR Map - the main use of this map is as a texture in ***Environment*** channels. With the Browser button one can select the ***HDRI*** map. ***VRay*** recognizes both the ***.hdr*** and ***.rad*** extensions.



■ **Figure 5.233**
Setting of the ***HDRI*** path

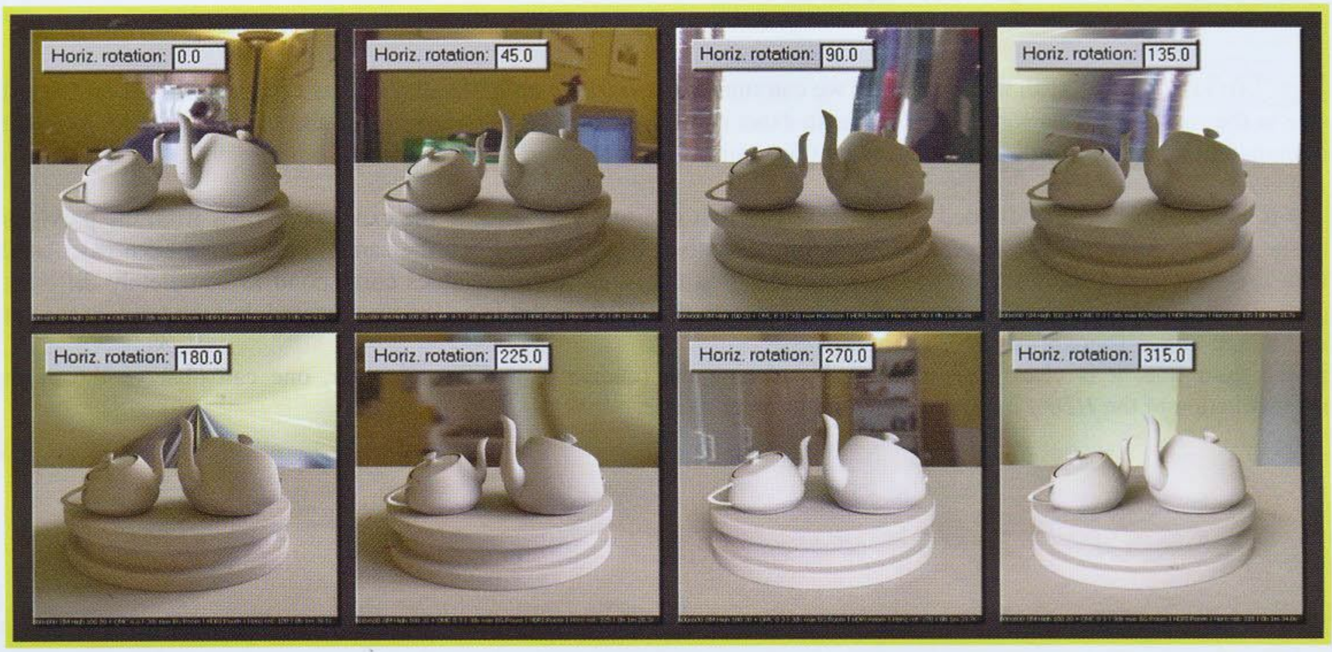
Multiplier - this spinner modifies the intensity of the ***HDRI*** map and therefore varies the amount of illumination. When set to 1.0 the map generates the default amount of light. With values below 1.0, the ***HDRI*** is darker and produces less light. Vice versa, the higher the value of the ***Multiplier*** parameter, the greater the intensity of light is.



■ **Figure 5.234**
Multiplier parameter being changed.

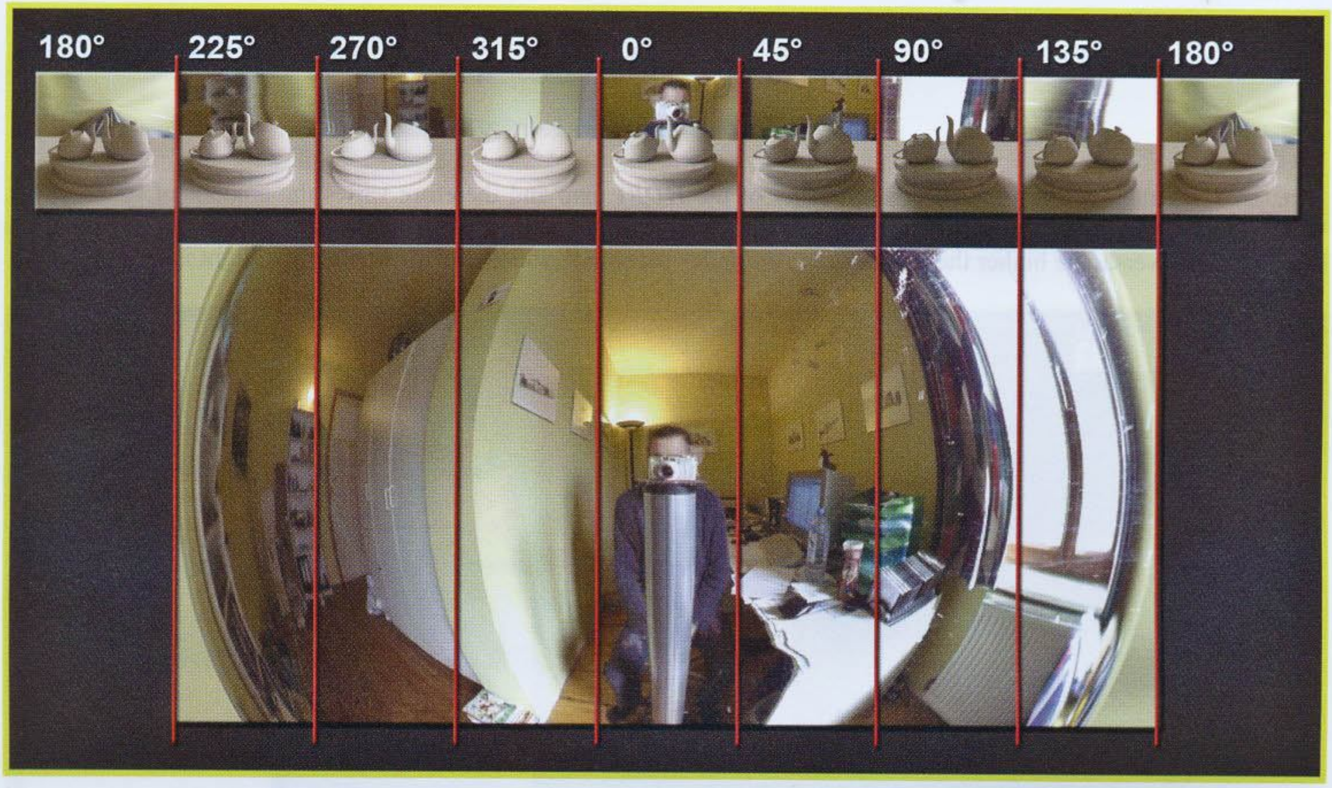
By changing the ***Multiplier*** one can see how both the illumination and the Background decrease or increase in intensity. This is because the map in the Environment of 3ds Max and ***VRay*** is instanced. Moreover, as luminosity increases, so does rendering time due to the greater amount of energy in the scene.

Horiz. rotation - ***HDRI*** maps are usually spherical. Thanks to this parameter one can rotate the map around its vertical axis, both left and right, as one might do with a spinning top.



■ **Figure 5.235**
Illumination and ***Background*** change according to the rotation angle.

Now analyze the ***HDRI*** in the ***Mirrored ball*** format and pay attention to the correspondence of the background with the ***HDRI*** portion of the image.



■ **Figure 5.236**
Correspondence between Background and the illumination referred to the degree of rotation of the ***VRayHDRI*** map.

In this image one notices one thing in particular: the ***HDRI*** and the background mirror each other. This may seem strange but if we think of how an ***HDRI*** is created, it is clear to see that this is absolutely correct.
Look at the sphere and imagine the photographer in front of the sphere. The camera is gripped on the right side and in the image it is correctly mirrored. The ***HDRI*** is used to map an imaginary sphere, within which the objects and geometries are placed. In other words, by observing the renders it is as if we take the place of the sphere and look around ourselves. If we look at the photographer from the sphere's point of view, the camera grip is on the left, just like in the render with ***Horiz. Rotation*** = 0.0. This could cause a bit of confusion and it would be easier if what we saw in the ***HDRI*** image, which can be visualized with HDRShop or Photoshop, were shown also in the 3ds Max Background. This is the reason for the ***Flip horizontally*** option.

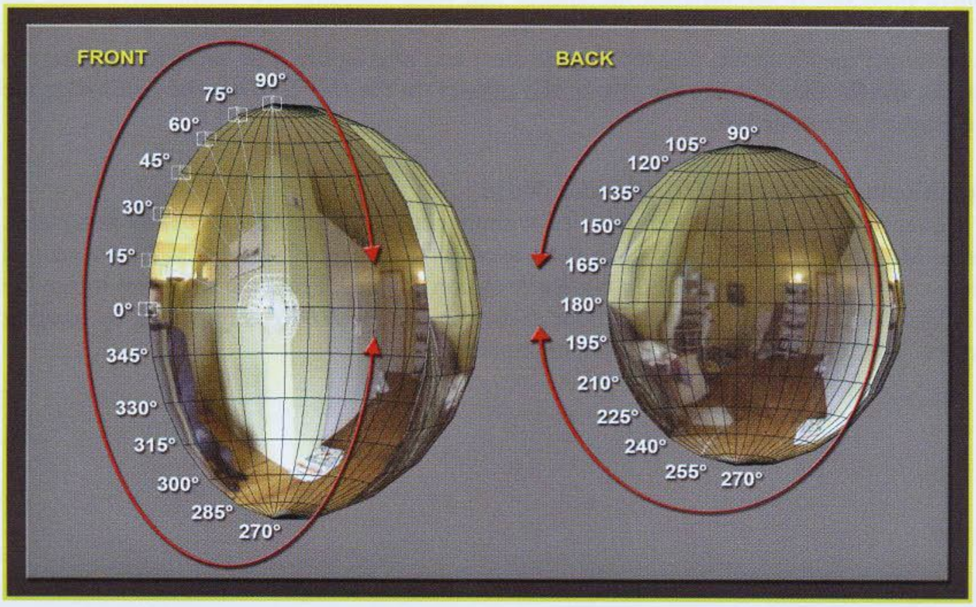
Flip horizontally - with this parameter we can mirror the ***HDRI*** map horizontally, thus bringing what is on the left side to the right side. This way we will have an exact correspondence between the ***HDRI*** and the background. Also the illumination in 3ds Max will be influenced by this modification.

The following image is subdivided into 4 lines. The first is a series of renders, the exact same ones from the previous image, created with the ***Flip horizontally*** option not active. In the second line the same images have been calculated with the ***Flip horizontally*** option active. It is clear that the same rotation values correspond to completely different portions of ***HDRI***. This time Background is aligned with the ***HDRI*** and it is no longer mirrored, but the renders are not in the right order. There is certain logic in this.
In the third line, it is clearly visible how by placing the renders in a different order, one can observe the exact reconstruction of the ***HDRI*** image reflected in the steel sphere.

■ **Figure 5.237**
Effects due to the activation of the ***Flip horizontally*** parameter

To sum up: in order to use an ***HDRI*** as a background image and preview exactly what will be rendered without having to proceed by trial and error, one must activate the ***Flip horizontally*** option and use the degrees shown in the third line. This system is valid only for ***Mirrored ball*** or ***Angular*** type ***HDRI*** images which are also the easiest to generate. Instead when using ***Spherical*** maps, the same concepts explained so far are valid, but the way the ***Flip horizontally*** and ***Horiz. Rotation*** parameters are applied is slightly different.

Vert. rotation - just like horizontal rotation, it is possible to flip the ***HDRI*** map vertically as well.



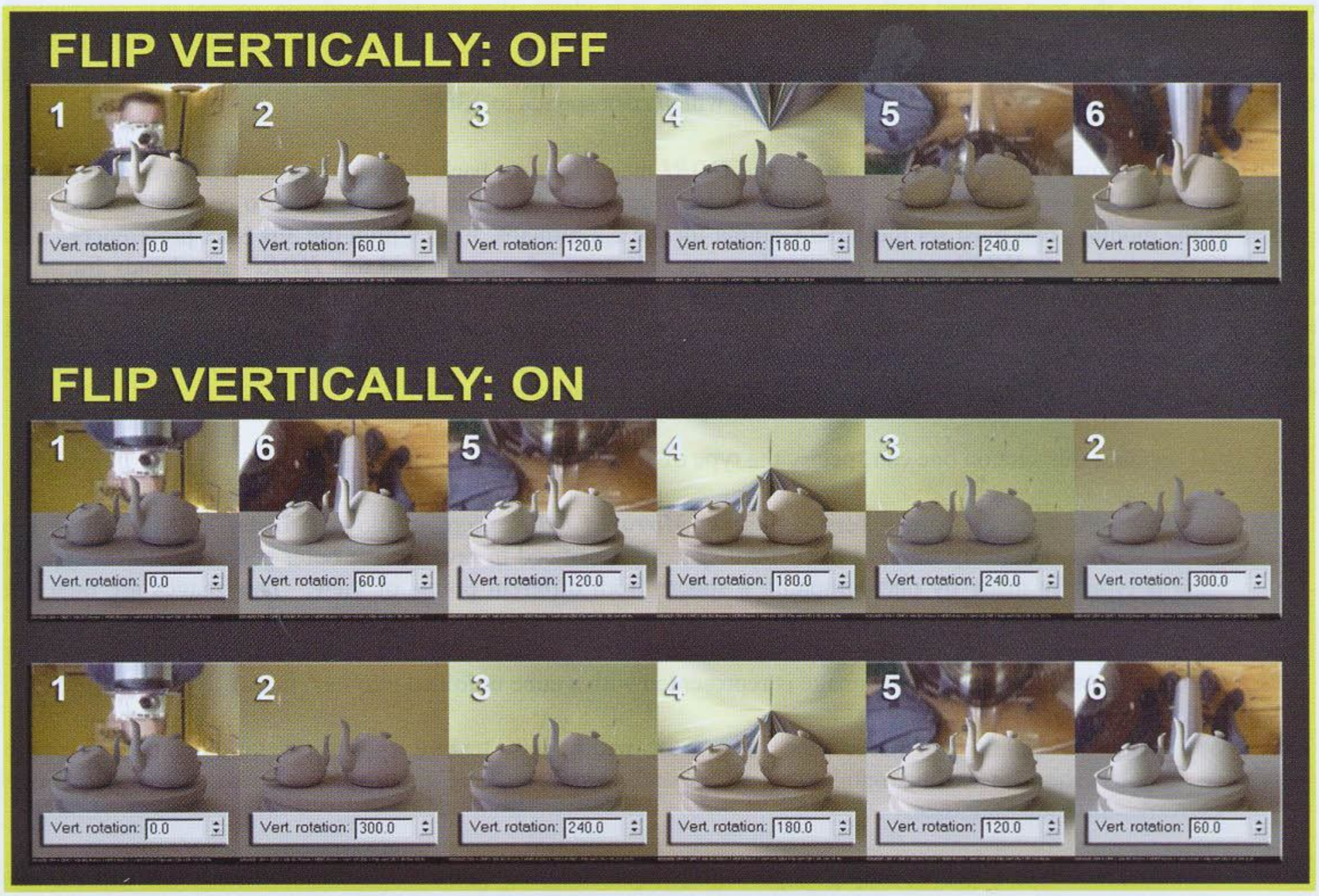
■ **Figure 5.238**
The front and back view of a sphere with its rotation degrees. It is not the camera which moves but the map which rotates. If one rotated the ***HDRI*** vertically by 30° with the ***Vert. rotation*** parameter, it would not be the camera to lift its trajectory but the ***HDRI*** to rotate downwards by 30°.

In the previous image a drawing explains the way *VRay* interprets the rotation angles of vertical rotations of ***HDRIs***. Basically, what is seen as the background in the render is what one would see from the chromed sphere's point of view.

Figure 5.239
Sequence of renders made by altering the ***Vert. rotation*** parameter.

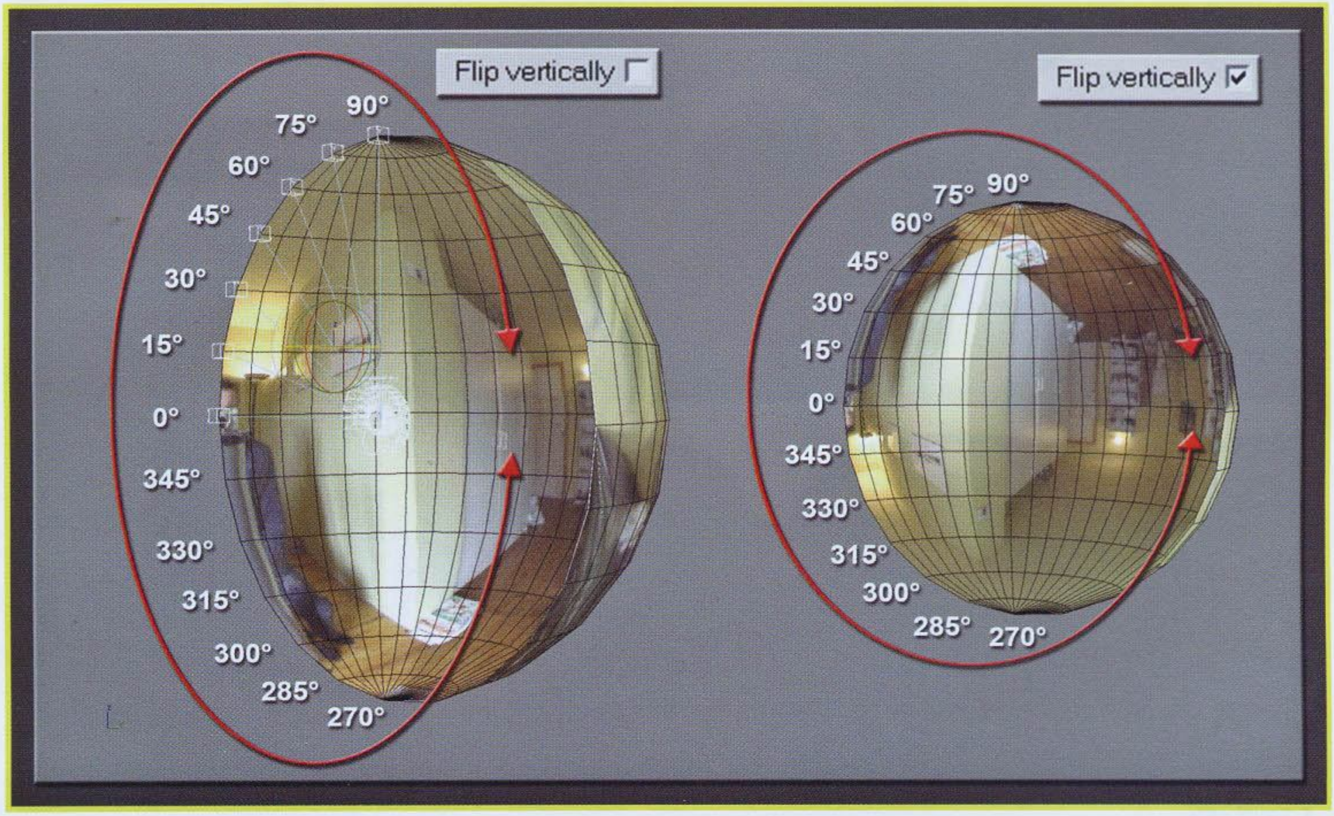
If we take a close look at this rendering sequence, for the first 90° of the map it is quite predictable. Again, if one imagines being in the place of the sphere, one simply moves ones line of view upwards, until it is perpendicular to the ceiling. In fact the render with ***Vert. rotation*** = 90 portrays the ceiling. By increasing the degrees of rotation, the virtual camera turns even more and starts to be upside down. The bizarre image which we find at 180° is none other than the photo of the spot which the sphere rests on. If for instance the sphere had been hanging, we would have seen the ground.
Beyond 180°, the background is no longer upside down, and the is quite clear and intelligible. The background image is in any case mirrored compared to the ***HDRI***. One can see the grip of the camera. In the ***HDRI Mirrored ball*** image the grip is on the right whereas in the render it is on the left.

Flip vertically - by activating this option we turn the ***HDRI*** upside down.
In the following image, the first sequence of renders has been fulfilled without activating the ***Flip vertically*** parameter. This series is used for comparison. Multiples of 60° angles have been used for the sake of simplicity. In the second sequence we have the same renders and angles but with the ***Flip vertically*** option enabled. From the very first image the ***HDRI*** is upside down. Now the ceiling is at the bottom and the floor at the top. As the ***HDRI*** is mirrored, by rotating vertically by 60°, what would we see? Obviously the feets of the photographer.



■ **Figure 5.240**
Effects of the ***Flip vertically*** parameter being activated.

The following image should solve any doubts.



■ **Figure 5.241**
Effects of the ***Flip vertically*** parameter being activated.

Carefully observe the map applied to the right-hand sphere. It is identical to the left-hand one, except it is overturned. By changing both maps by 60°, that is, by rotating the by 60° downwards, in the left-hand sphere a portion of the ceiling appears, instead in the right-hand one the photographer's feet can be seen. In order to make the ceiling visible also in the right-hand sphere, one must rotate by -60°, which corresponds to +300°.

Gamma - this allows one to set the gamma of the ***HDRI*** map. If we go back the odd chapter when we talked about LWF, in order to linearize .jpg images characterized by a gamma which has been modified to contrast the inverted gamma of the monitor, one must set the Input Gamma for the 2.2 bitmaps, from within the 3ds Max preferences. This way all images, including ***HDRIs*** are automatically corrected. But we have also said that ***HDRIs*** are already linear, in other words they do not have a gamma which is inversely modified like LDRIs. For this reason, if there was not the option ***Gamma*** for ***HDRI*** images which allows one to specify which gamma to assign to maps, the ***HDRIs*** would undergo an unnecessary alteration, which would in fact be damaging. By default the gamma is 1.0 and should rarely be changed unless one is handling HDRI maps with a modified ***Gamma*** which is not 1.0.

Map type - within this section one can specify the type of ***HDRI*** map used.

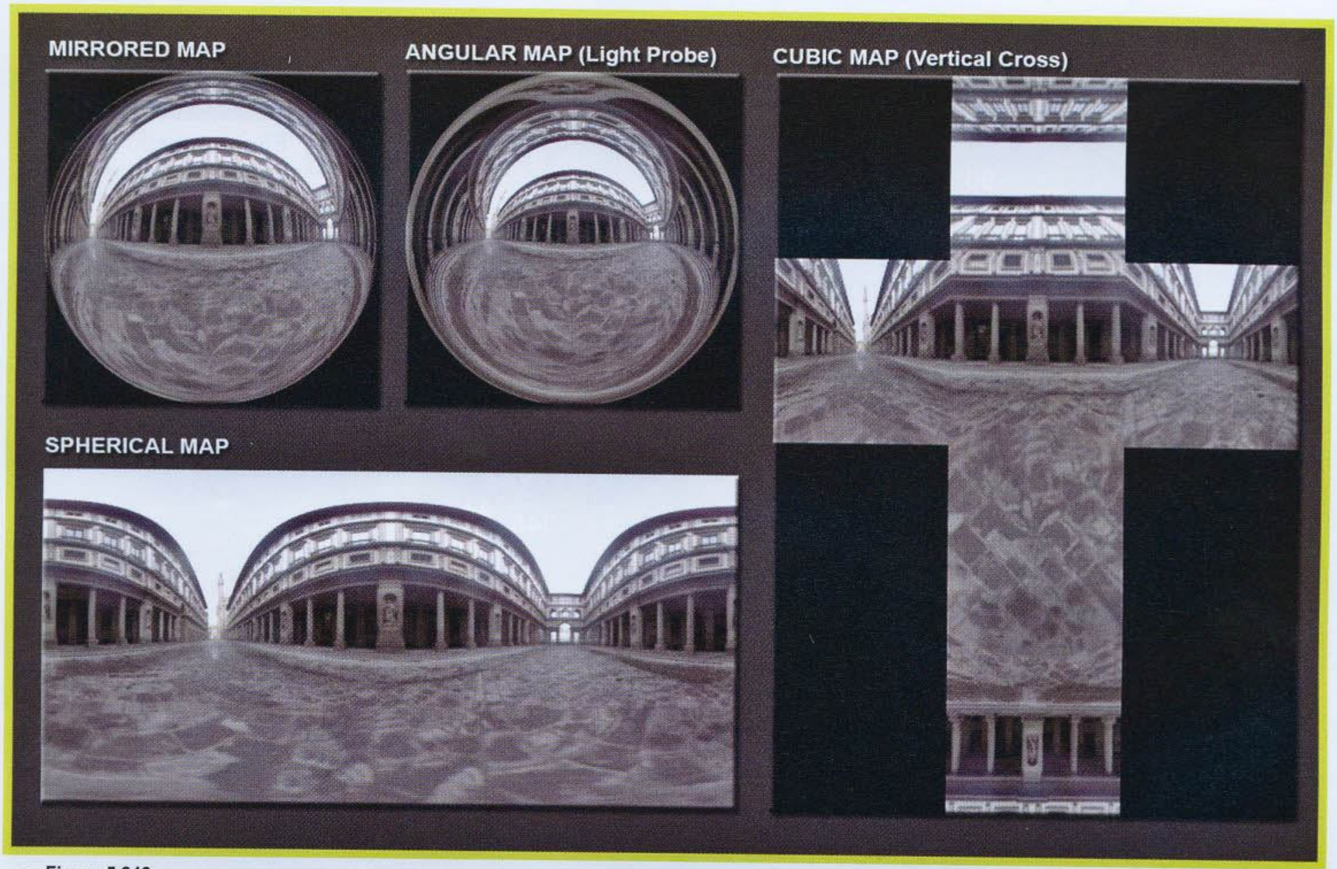
- ***Angular***: this type of map is very similar to ***Mirrored ball***, except it allows a better quality around the circumference. Basically some of the central space is sacrificed in favor of the outer geometries. This type of map is typically made by photographing a sphere from different points of view. The images are then put together in an ***HDRI*** of the ***Angular*** type.

- ***Cubic***: this strange type of 360° image is represented with a cube unfolded into its 6 faces.

- ***Spherical***: the classic panoramic map, where the entire 360° texture appears as unrolled onto a single rectangular sheet. The horizontal size corresponds to the sphere's latitude, whereas the vertical size corresponds to the longitude.

- ***Mirrored ball***: map obtained simply by photographing a chromed sphere.

- ***Explicit map channel***: this allows one to manually specify map coordinates.

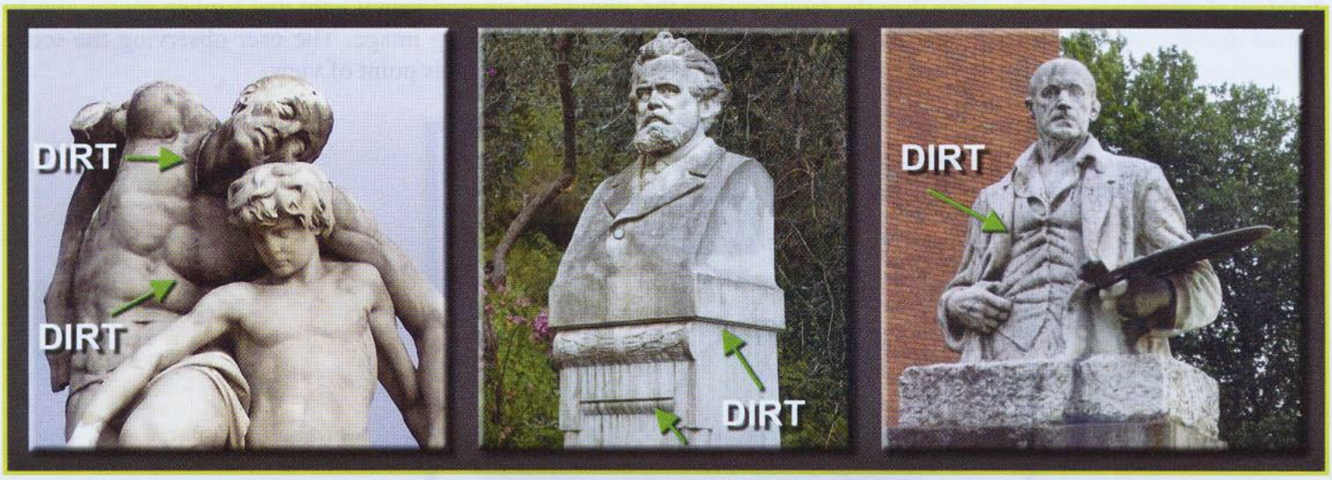


■ **Figure 5.242**
The four main types of ***HDRI*** maps. Notice how in the ***Angular*** map the central porch is smaller than in ***Mirrored wall***, and lateral arches are, in proportion, much larger.

VRay Map: VRayDirt

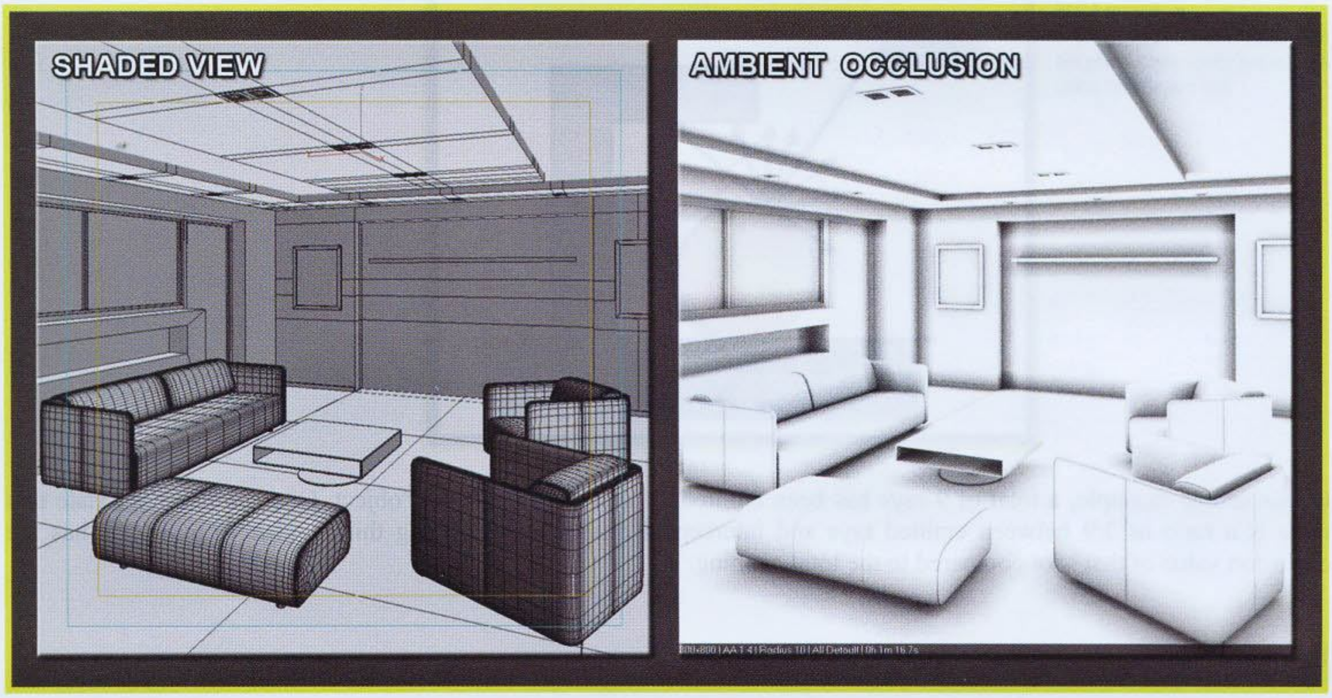
INTRODUCTION

This particular map allows one to simulate certain types of effects. For instance one can simulate the dirt that accumulates between the cavities of statues or objects, but more often it is used for simulating ***Ambient Occlusion***.



■ Figure 5.243
On complex surfaces, such as statues, inside cavities and such, dirt often accumulates. This phenomenon can be simulated by *VRayDirt* maps.

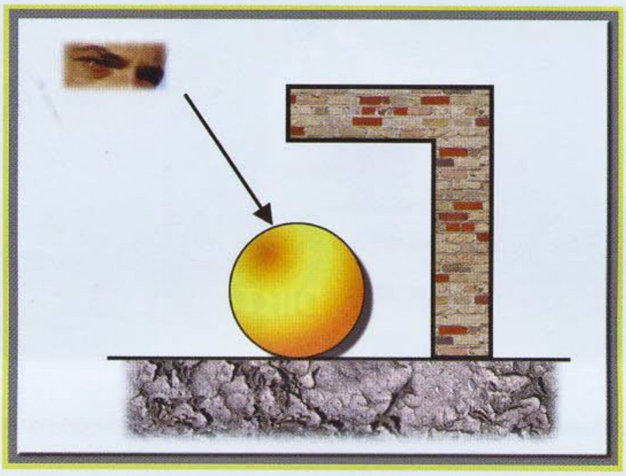
If the concept of dirt is quite simple, the same can not be said for the subject of ***Ambient Occlusion***, which is abbreviated as ***AO***.



■ Figure 5.244
VRayDirt map being used for generating *Ambient Occlusion*.

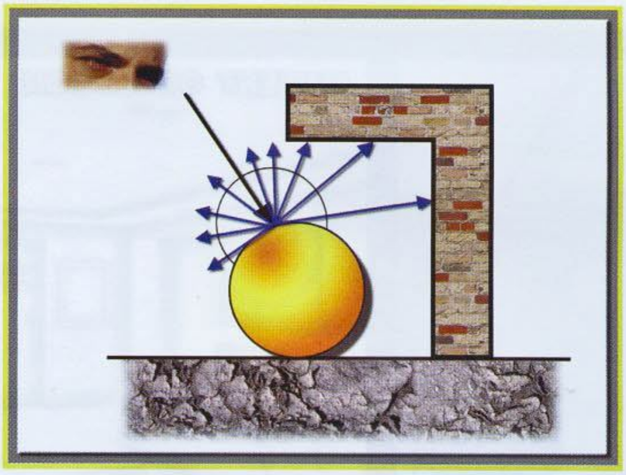
In the previous scene, no light source has been used, and yet it seems like something is illuminating the whole scene. The only adjustment made has been to associate all the objects in the scene to the same map, ***VRayDirt***. Not even the GI has been activated. How does ***Ambient Occlusion*** work?

The use of Global Illumination in the last few years has permitted mass-production of photorealistic renders. This has been possible also mostly thanks to ever more powerful computers. The real Achilles' heel of the GI is the machine time, which means the time needed for the computer to calculate all the bounces carried out by rays. If we move on to the subject of animations, things get even more complicated. Often one must invest a great amount of money in hardware in order to carry out a GI animation.
Ambient Occlusion has been used and still is, by the greatest CG studios for simulating the Global Illumination, without it actually having to be present. It is a way to obtain an a reasonably realistic render, especially suited for animations, where photorealistic quality is of secondary importance compared to animation quality, the true heart of 3D films.
Ambient Occlusion is nothing more than the ratio between the amount of environmental light that a given spot is able to receive and the total amount of light that it can receive. Observe the following image. The user observing the scene intersects any geometry which is visible to him with the rays which come from his point of view.



■ **Figure 5.245**
The first ray emitted by the camera.

After, from the point where the first rays meet the surface, a precise number of new rays are emitted randomly, in a radius-oriented and spherical manner.



■ **Figure 5.246**
Secondary rays emitted in order to analyze the space surrounding the first intersection point.

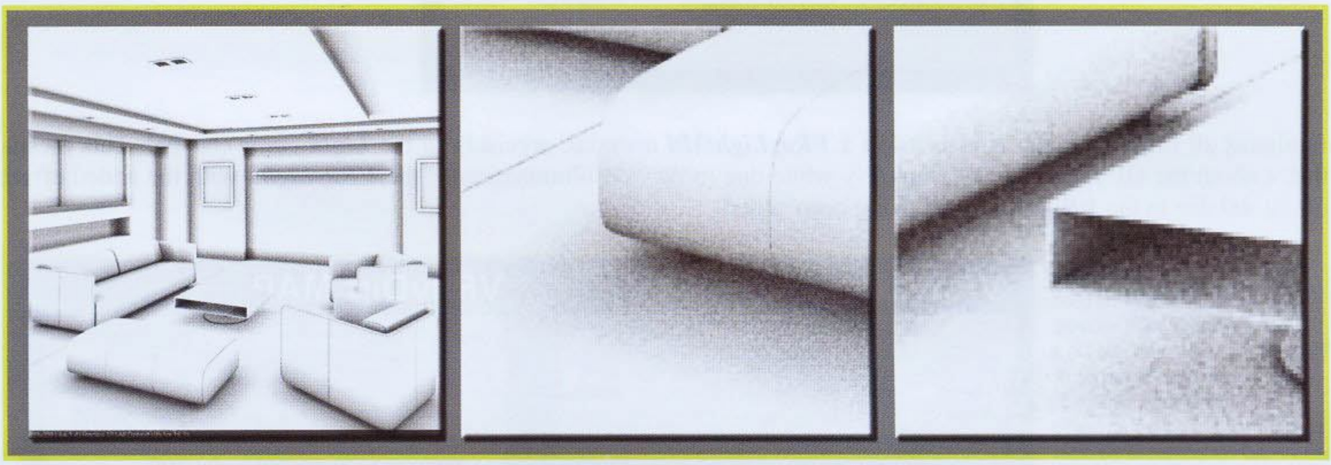
In this simple example, a total of 9 rays has been emitted, of which only 3 hit an object. One can therefore state that there is a ratio of 3/9 between emitted rays and intercepted ones. By subtracting this number from 1, one has the occlusion value of that spot compared to the total opening.

1- 3/9 = 0.666

Intersecting rays	Occlusion value
0	1.0
1	0.8
2	0.7
3	0.6
4	0.5
5	0.4
6	0.3
7	0.2
8	0.1
9	0.0

By associating a completely white tone to the value 1.0, a completely dark tone to the value 0.0 and different tones of grey to all the intermediate values, one can easily see how one obtains the ***AO*** value shown previously. So, for an actual render, the ***AO*** is a sort of analysis of the occlusion of some objects compared to each other. The more occluded portions of objects are darker whereas the freer parts are closer to white.

Taking a close look at the image produced via ***Ambient Occlusion***, one notices a certain grainy quality, which is typical of the ***QMC*** algorithm. The ***AO*** is produced without interpolation.

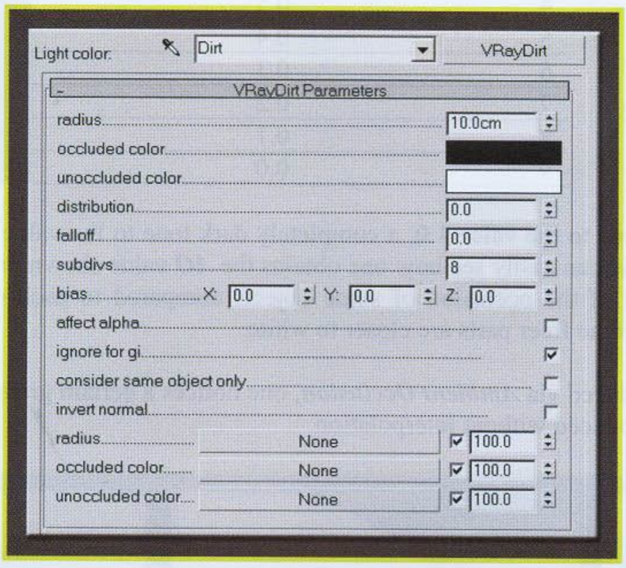


■ **Figure 5.247**
Grainy appearance and noise due to the low quality of ***AO*** sampling.

In actual fact, for ***AO*** calculation, a ***QMC*** system has been used for determining the occlusion value. Think of the last example. 9 rays were used for calculating the ***AO*** and this ensured a value "x" which was more or less close to the true value. If more rays had been used the occlusion value result would have been even closer to the truth, because the samples for calculation would have been more numerous. If we apply this concept to all the pixels in the image, we can imagine that the more rays there are, the more correct the result will be.
As we can see in the previous image, the ***AO*** has nothing to do with the GI, but its appearance reminds one of the way global illumination looks and this is why it is often used to simulate it.

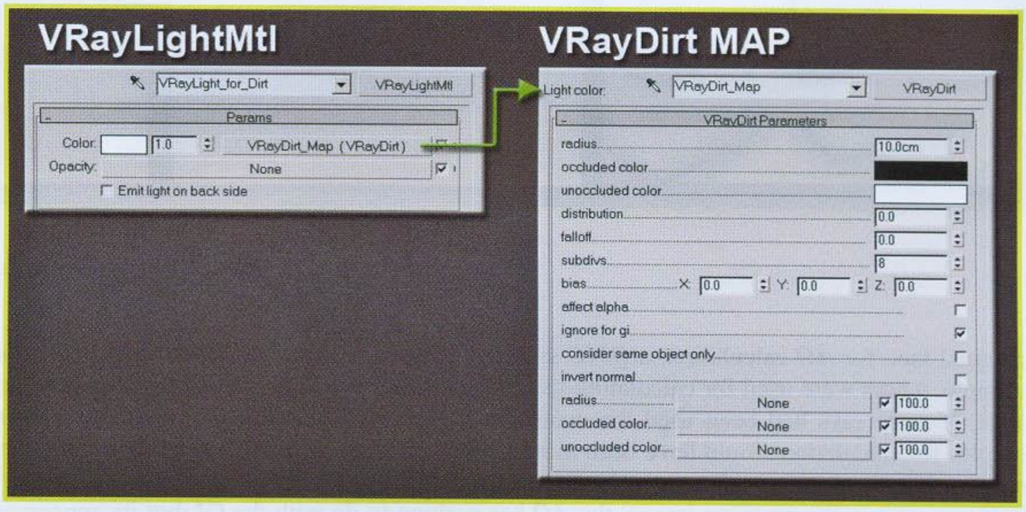
PARAMETERS

This particular type of map is different to others because it is not used for texturing (in the true meaning of the word) a mesh. It is used for generating a spatial analysis of the scene.



■ **Figure 5.248**
The panel containing all the settings for the ***AO***.

In almost all cases this map is inserted in a ***VRayLightMtl*** material, precisely in the ***Light color*** channel. This means that without the GI, the scene is completely white due to the self-illumination of the ***VRayLight***, with the added effect of the ***AO*** due to the ***VRayDirt*** map having been added.

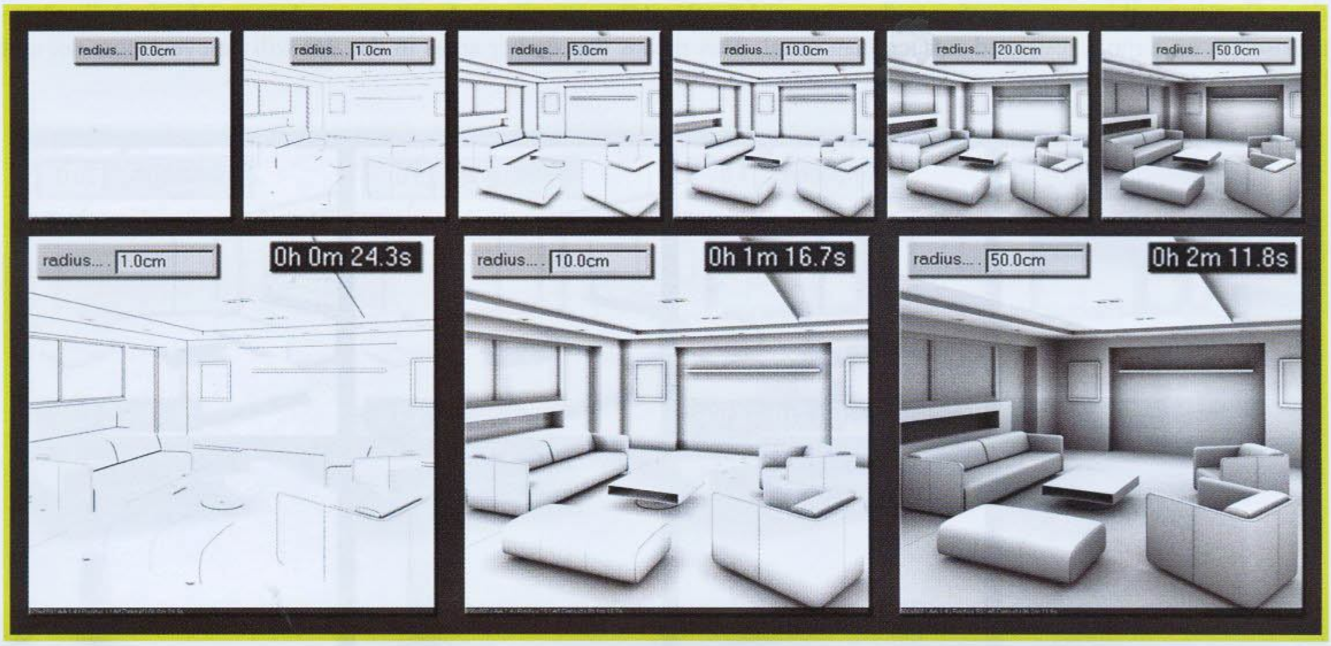


■ **Figure 5.249**
The ***VRayDirt*** map being used within a ***VRayLightMtl*** material. The render can be composited in a program such as Photoshop or Combustion and blended in multiply mode in such a way as to add the ***AO*** component to the main render.

Once the material has been created, one can assign it to all the objects in the scene via the Material Editor, or one can use the ***Override mtl*** function located in the ***VRay:Global switches*** rollout in the *VRay* Renderer.

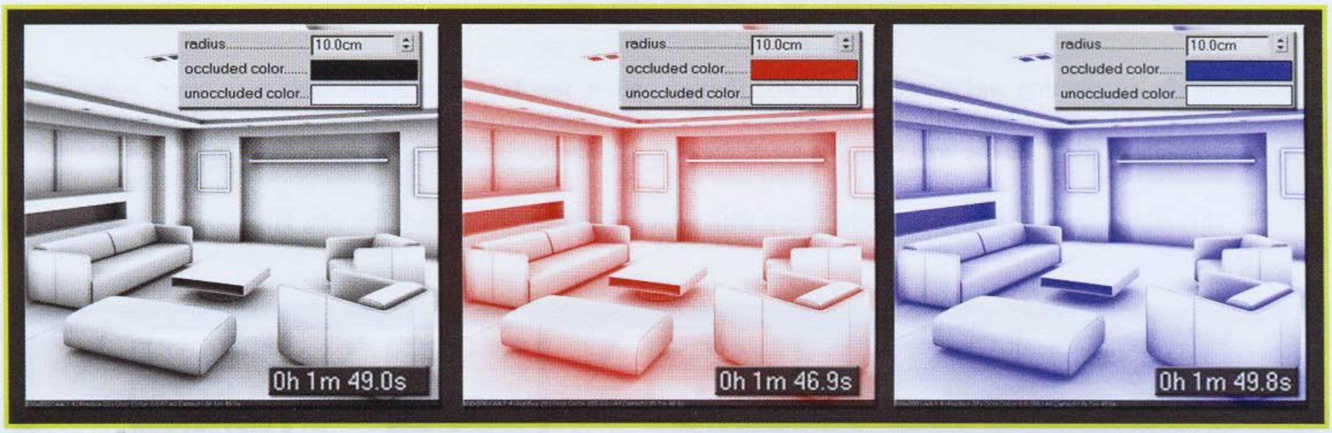
radius - this parameter allows one to determine the zone of influence of ***Ambient Occlusion***. The greater its value, the more it influences the ***VRayDirt*** effect. In order to control the radius parameter one can use a texture as well.

Some observations to be made on the following image. The first concerns the measurement unit to be used for the ***radius*** parameter. In this file we have used the cm unit: the scene is therefore perfectly in scale, so it is very easy to know the value of the ***AO*** radius. The second observation concerns the influence of ***Ambient Occlusion***. With high values dirt tends to expand. For very high values, like 50.0 cm, the effect is similar to that obtained with Global Illumination, which is the main purpose of using the ***AO***. Lastly, by analyzing rendering time, it is correct to state that the greater the range of action of the ***AO***, the longer it will take to render. This is because *VRay* must use a greater number of samples on a vast surface and trace a greater number of rays, with as a result an increase in the number of calculations to be carried out.



■ **Figure 5.250**
The ***radius*** parameter being changed. The ***AO*** expands.

occluded color - this indicates the color to be used for ***Ambient Occlusion***. For its determination it is also possible to use a texture.

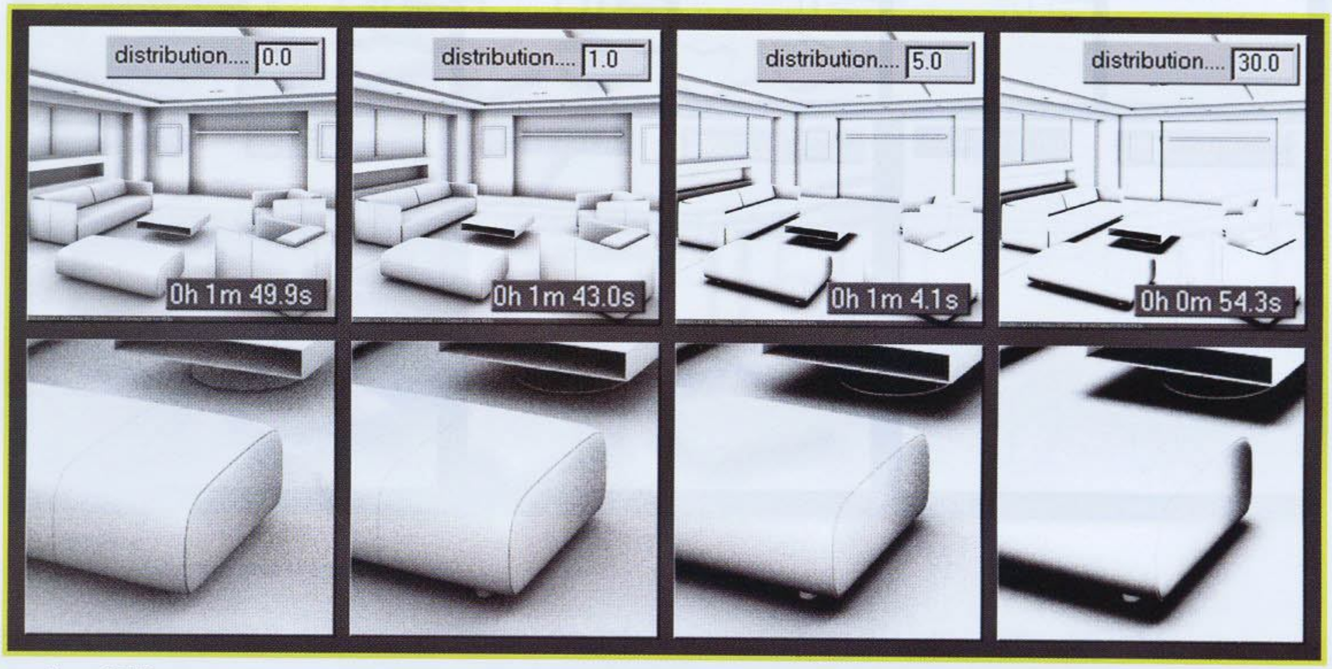


■ **Figure 5.251**
Variazione del colore dell'***AO***.

unoccluded color - this indicates the color to be used for the areas which are not interested by ***Ambient Occlusion***. For its determination one can also use a texture.

■ **Figure 5.252**
Variation of the complementary color of the ***AO***.

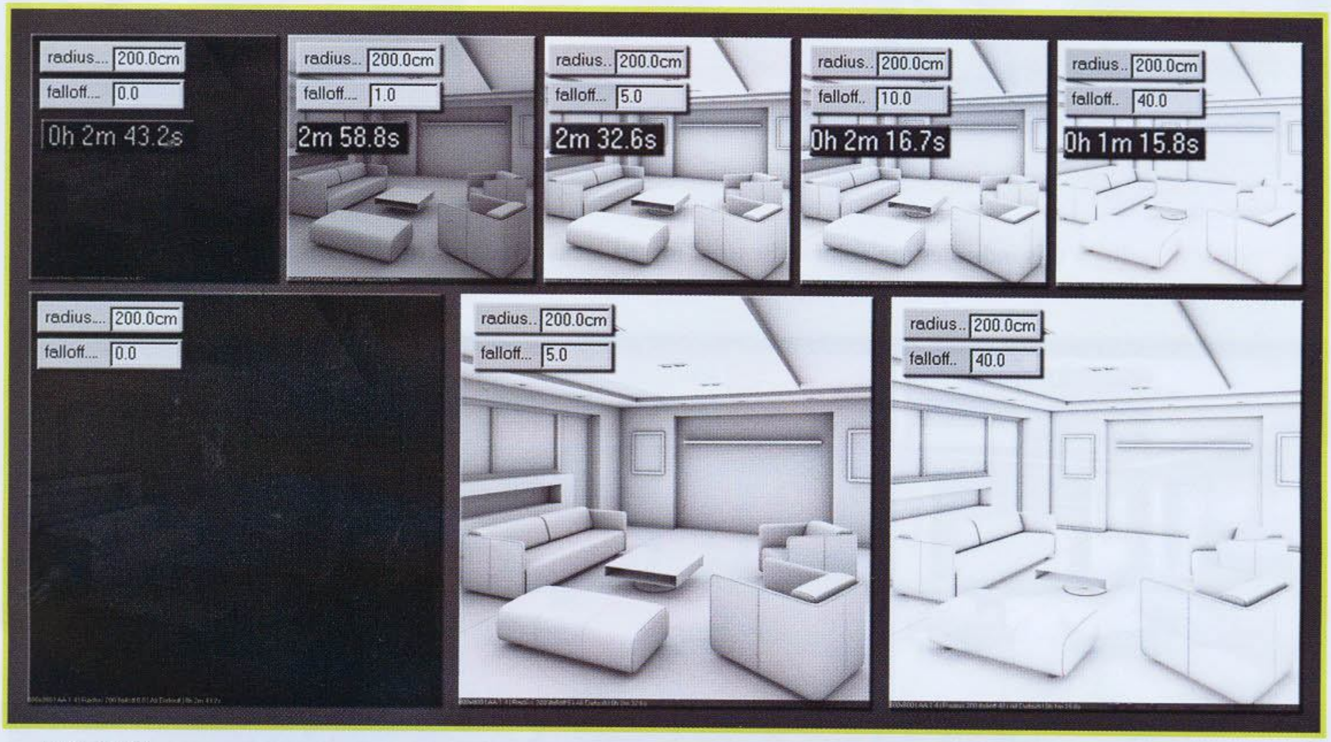
distribution - this parameter forces the rays used for the ***AO*** to remain as closely grouped as possible and distribute themselves on a more restricted surface than usual. This means that darker areas in the ***AO*** will be even more so, and brighter areas even brighter.



■ **Figure 5.253**
The ***distribution*** option is useful if one wants to further emphasize the occluded areas.

By reducing the distribution and increasing the ***distribution*** option, the ***AO*** concentrates itself even more, reducing the gentleness of the blend one has with default settings. ***AO*** can be emphasized by increasing the ***distribution*** parameter. Moreover, having diminished the area of influence of the ***AO***, also rendering time goes down drastically.

falloff - this allows one to specify the immediacy with which ***Ambient Occlusion*** dies out. High values make the ***AO*** finish sooner.



■ **Figure 5.254**
Definition of the speed of propagation of the ***AO***.

In the previous example a very high ***radius*** was used. It could seem like an excessively high setting, as we can see from the first image in the previous representation, where the environment has come out completely black. This is where the ***falloff*** parameter kicks in. With ***falloff*** = 1.0, the environment is dark but clearer compared to the previous test and above all, it has a softer AO. With value 5, we return to a good level of brightness, with a very smooth ***AO***. With a value of 40, the ***AO*** dies out soon although gradually. In this image we can see the difference compared to a ***radius*** = 10 without ***falloff***.

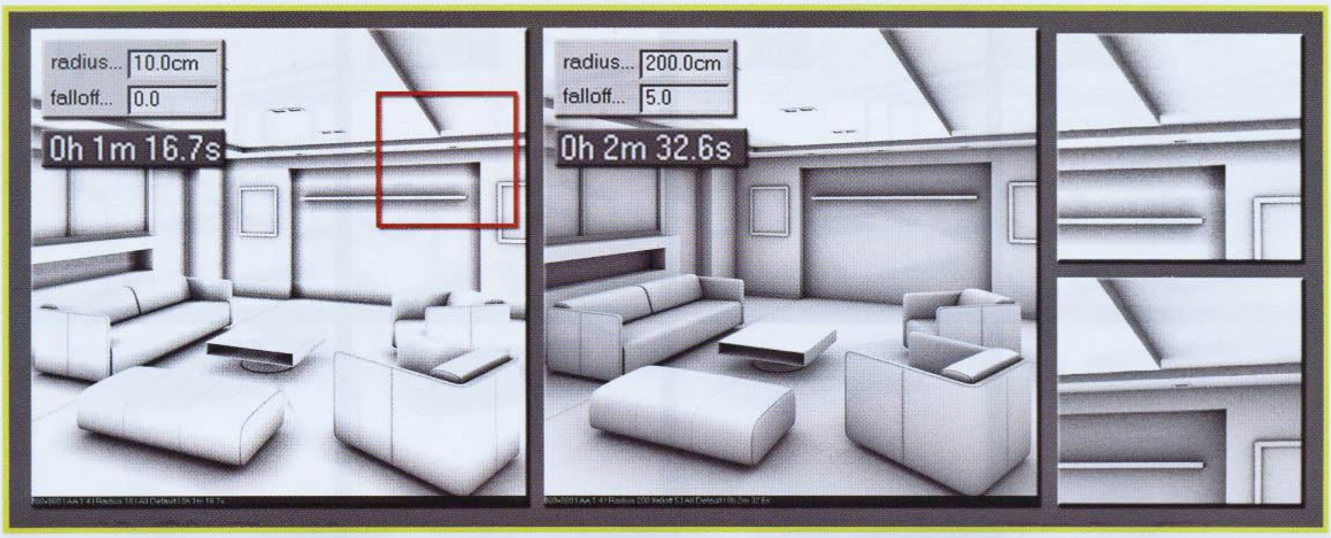


Figure 5.255
Variation of ***AO*** color.

In this direct comparison, the starting color in the occluded zones of the ***AO*** is similar, but not the way it propagates. In the first case, with default settings, the ***AO*** finishes very brusquely, almost as if its limit had been outlined. In the second case this phenomenon is not present anywhere in the scene. Obviously rendering time suffers because with ***falloff*** = 5 there are less areas which are completely white and without samples.

subdivs - the square of this value is equal to the number of samples used for ***AO*** calculation.

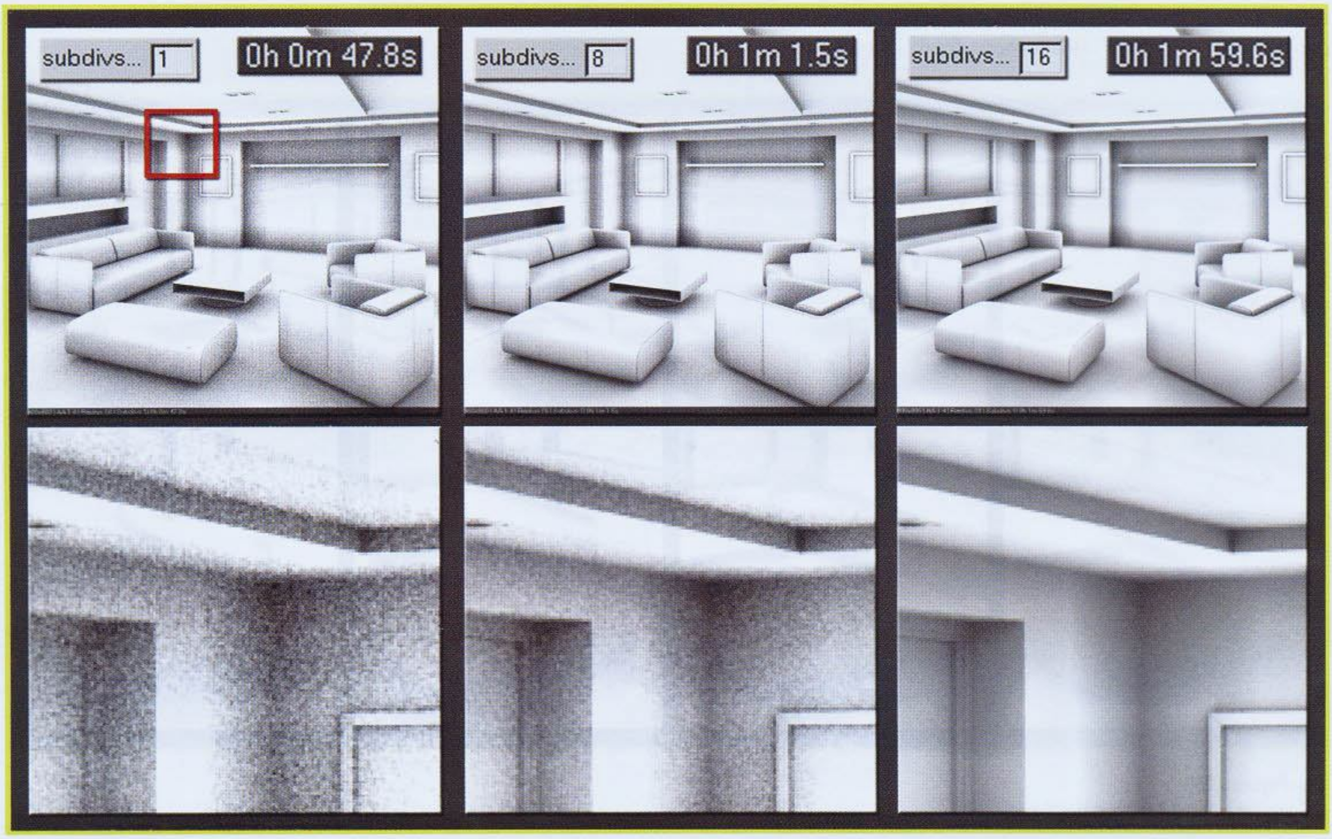
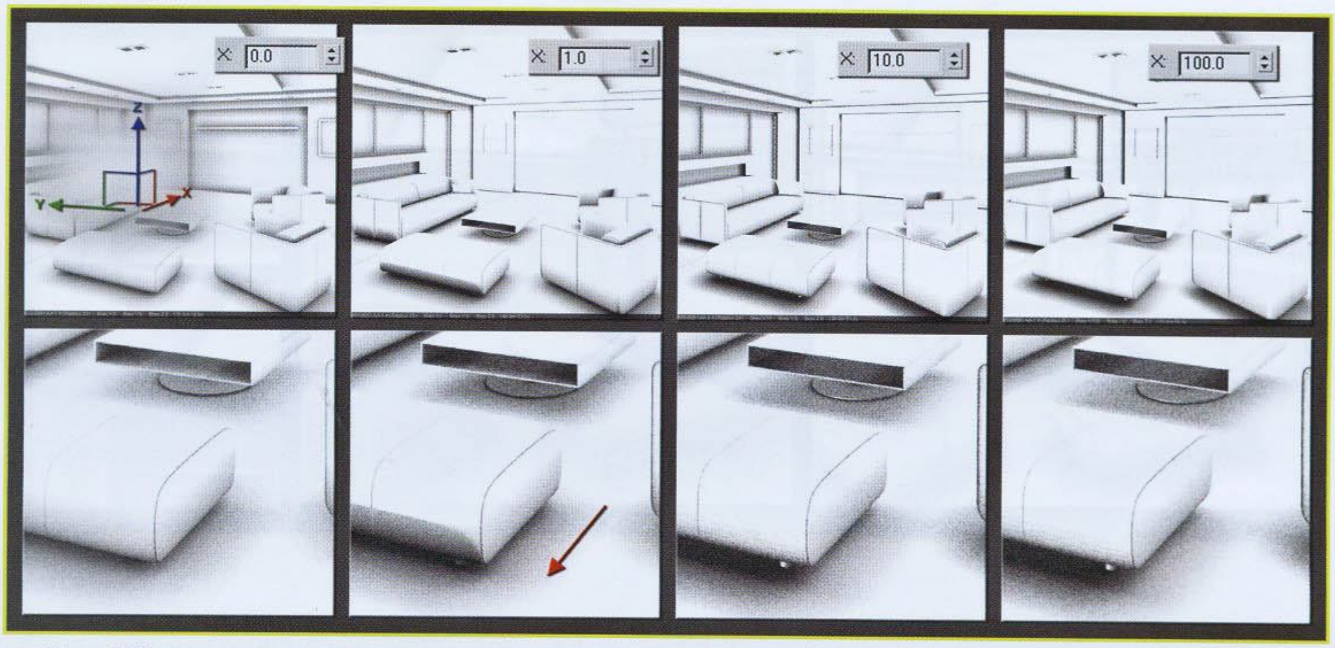


Figure 5.256
Quality vs. rendering time.

The higher its setting, the more rays are emitted and the better quality is at the cost of higher rendering time. From the examples one can see how time increases drastically, more than double when using high settings, but so does quality improve significantly. With ***subdivs*** = 16 an adequate level of quality has been obtained within acceptable times.

bias (X, Y, Z) - with the use of these parameters it is possible to control the direction of the ***AO*** calculation along the three axis' X, Y and Z.

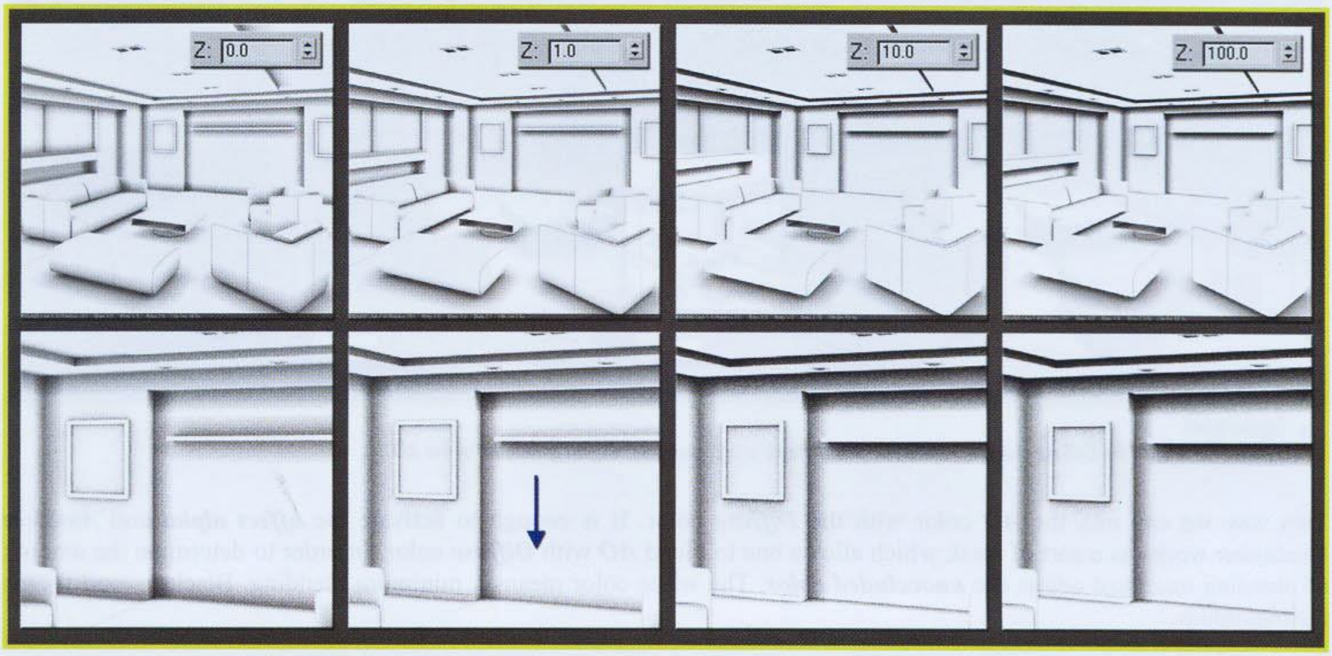


■ **Figure 5.257**
Translation of the ***AO*** along the X axis.

By increasing the ***bias x*** parameter one can translate the ***Ambient Occlusion*** along the X axis, as shown by the gizmo in the image. It can be seen in the area under the sofa, where the dark part tends to increase. Unfortunately the camera is almost on the same axis as X and so it is not simple to notice the effect of this parameter.
The same thing can be done with the Y axis. In this case, as the camera is orthogonal compared to the Y axis, the ***AO*** movement appears more clearly. If we take a look at the ***Ambient Occlusion*** under the sofa, as in the previous case, one can see the "shadow" movement from left to right.

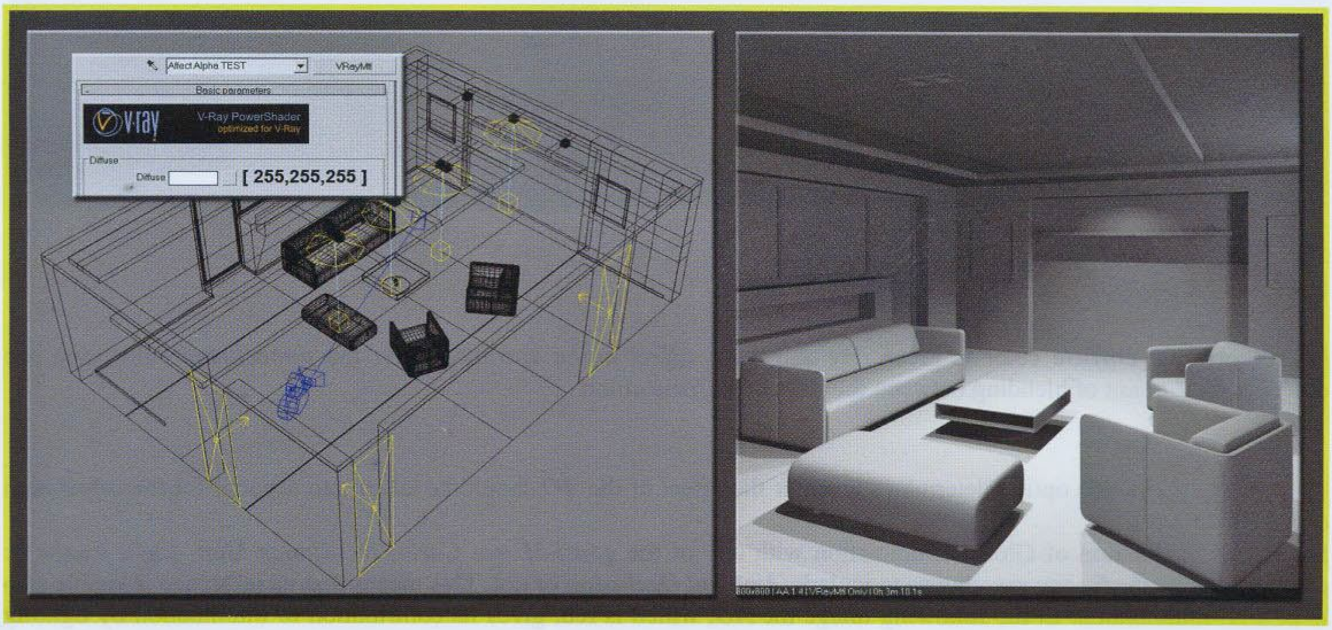
■ **Figure 5.258**
Translation of the ***AO*** along the Y axis.

And finally, the Z axis. In this case the ***AO*** goes from top to bottom. It can be observed clearly near the shelf and along the walls. Whereas with setting 0.0 the shelf is uniformly surrounded by occlusion, by increasing the settings, the ***AO*** is "squashed" along the vertical position.



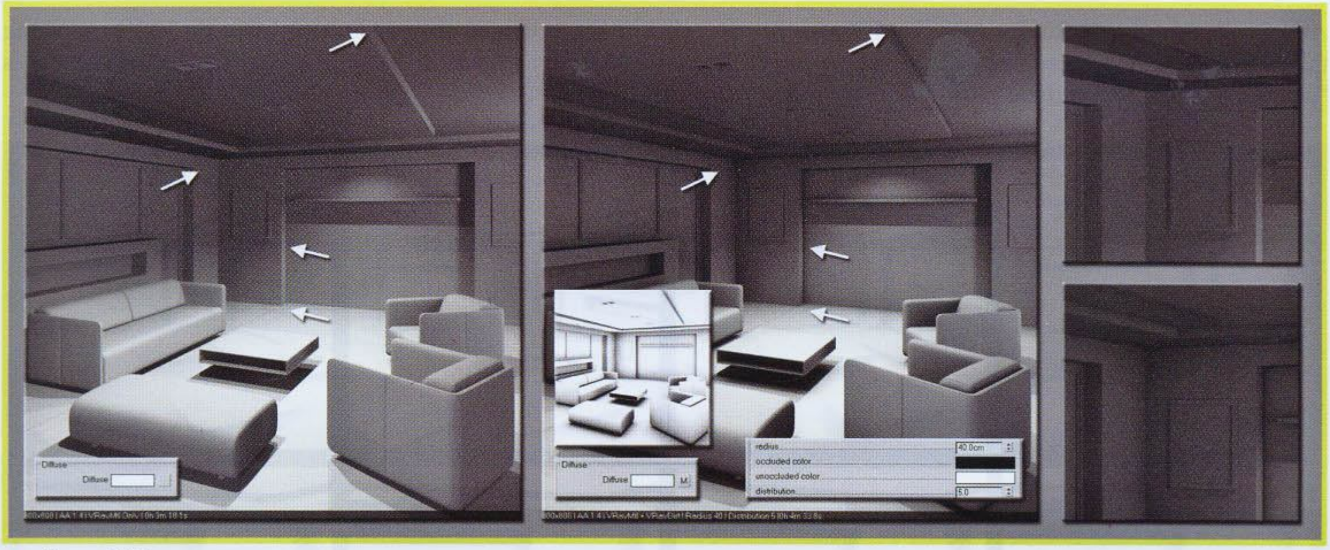
■ **Figure 5.259**
Translation of the ***AO*** downwards.

affect alpha - this parameter allows one to blend the ***VRayDirt*** map, in this case, with the color in the ***Light color*** channel. For example: if the ***VRayDirt*** map had been inserted in the ***Diffuse*** channel of a ***VRayMtl***, activating the ***affect alpha*** parameter would have allowed the blending of that map with the color in the ***Diffuse*** channel.
In the following scene, rendered with no GI, four Spotlights have been activated and three ***VRayLights***. A light grey ***VRayMtl*** has been assigned to all geometries.



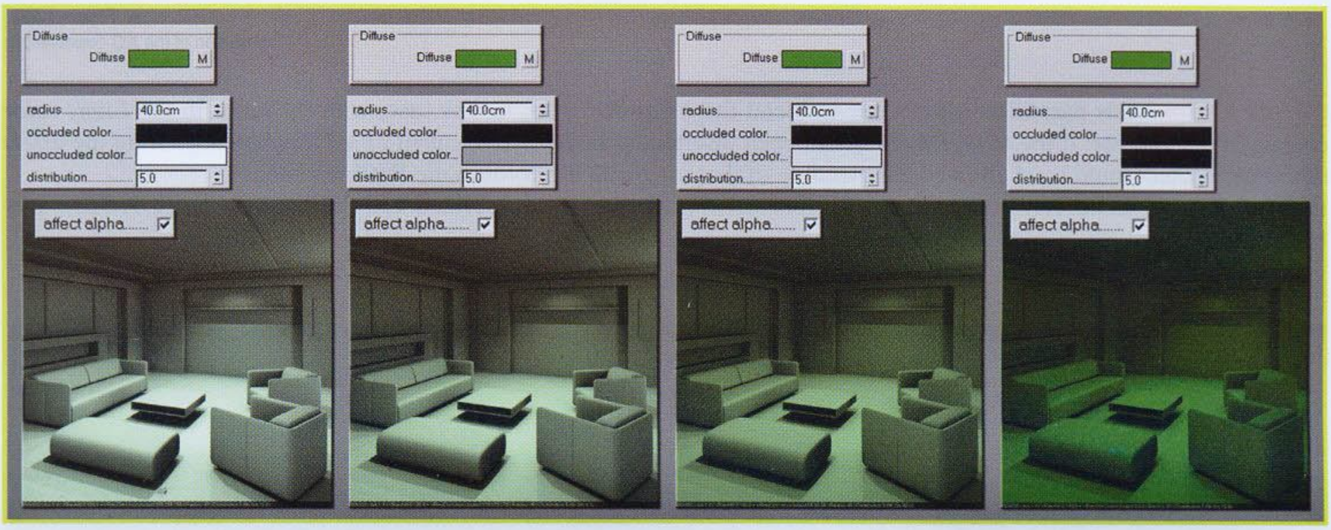
■ **Figure 5.260**
Traditional illumination without Global Illumination by means of seven light sources.

What happens when one substitutes the diffuse color with ***VRayDirt***, as the ***unoccluded color*** is RGB = [255, 255, 255], is that the whole scene has dirt added to it without any other chromatic variation.



■ **Figure 5.261**
VRayDirt map used in the ***Diffuse*** channel. Notice the greater depth of light along the edges, generated by the ***AO***.

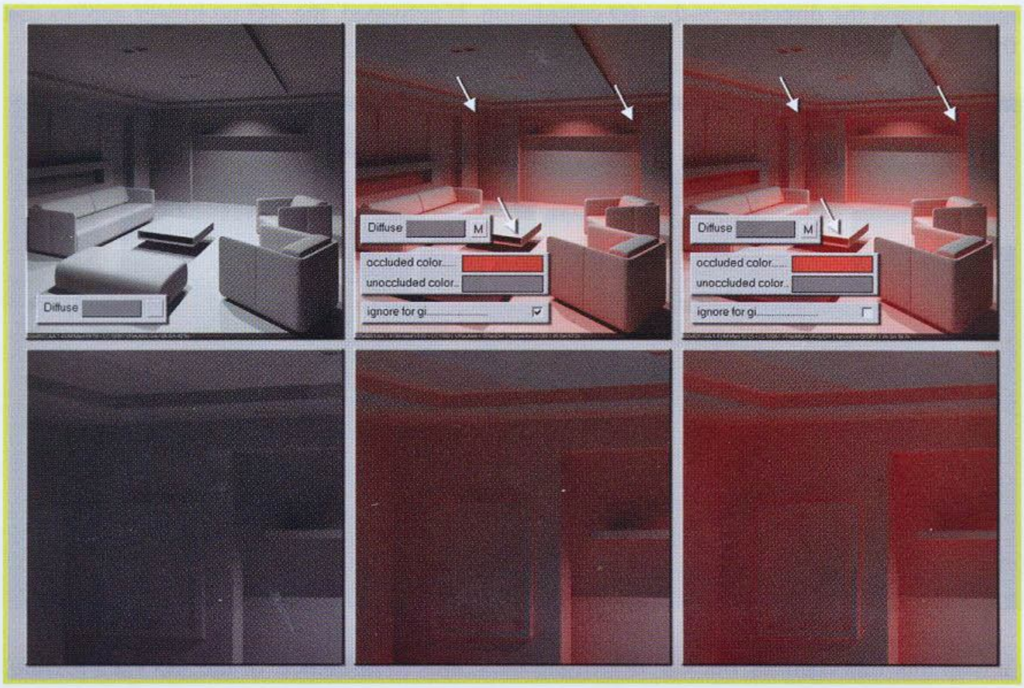
This way we can mix the ***AO*** color with the ***Diffuse*** color. It is enough to activate the ***affect alpha*** and ***Ambient Occlusion*** works as a sort of mask which allows one to blend ***AO*** with ***Diffuse*** color. In order to determine the amount of blending one must adjust the ***unoccluded color***. The white color means a minimum blending. Black instead means the maximum.



■ **Figure 5.262**
Blending between ***Diffuse*** color and ***AO***.

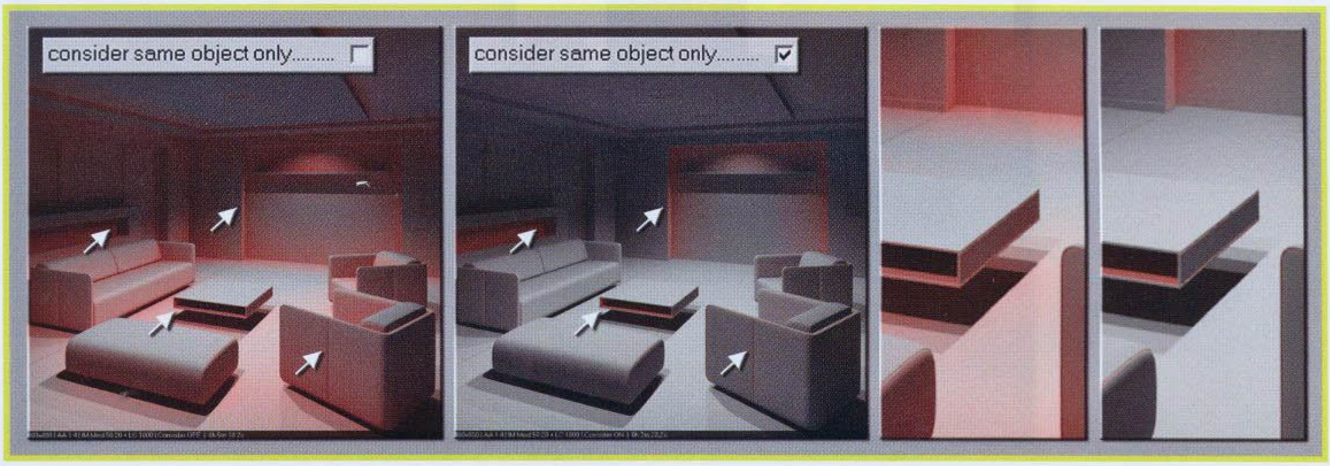
In this case the ***Diffuse*** color is green. By activating ***affect alpha*** and modifying the ***unoccluded*** color, one can establish the amount of blending between the ***AO*** and ***Diffuse*** color.

ignore for GI - this option determines whether the effect of the ***AO*** should be taken into account for the calculation of the GI.
During the calculation of Global Illumination with any of the ***quasi-Monte Carlo***, ***Irradiance Map***, ***Light Cache*** or ***Photon Map*** methods, one can choose to include ***Ambient Occlusion*** or not. This means that its influence is visible also in general illumination. In the following image the ***AO***, which is red, gives its contribution to illumination by lighting the areas influenced by its presence. If this parameter were not activated, as is so by default, the red of the ***AO*** is not calculated and for GI calculation only the ***unoccluded color*** is held in consideration.



■ **Figure 5.263**
Effects of the ***AO*** on the Global Illumination.

consider same object only - when this is active, the ***dirt*** effect is calculated exclusively on objects, without considering contact surfaces. When it is deactivated, all the geometries in the scene which are directly connected to the ***dirt*** effect take part in the calculation of ***Ambient Occlusion***.



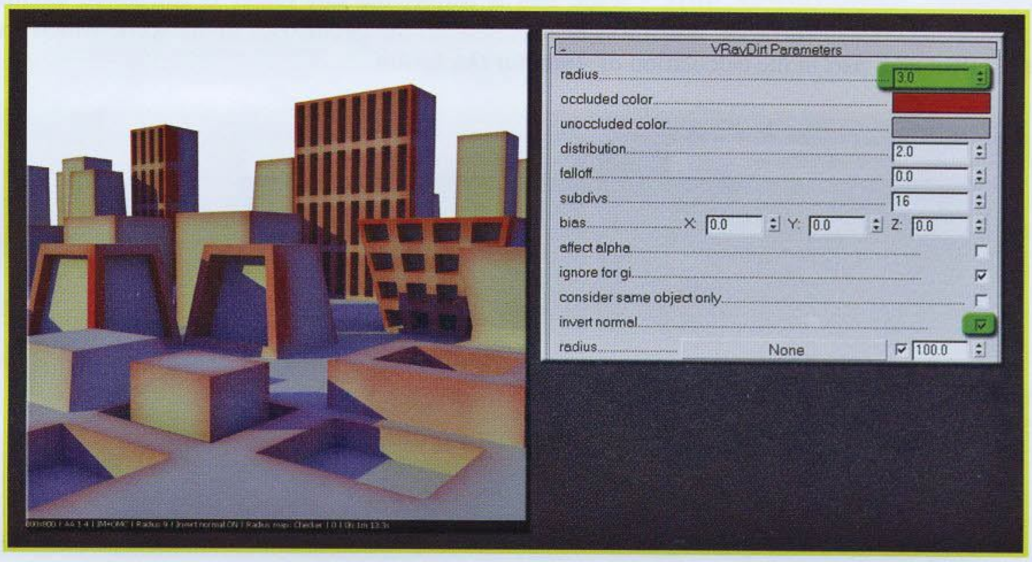
■ **Figure 5.264**
Using the ***consider same object only*** parameter limits the ***AO*** calculation to the object, without considering the relationship with other geometries.

By activating this option, the ***AO*** is only calculated on the object, as if it were an entity of its own and nothing were next to it. So for example, on the sofas the ***AO*** is calculated only along the stitching or on the walls only inside the nooks. This is because they are all separate geometries.
If for instance one decided to unite all the objects together in one mesh via the Attach command, the activation of the ***consider same object only*** parameter would not bring about any changes, because all the polygons would be part of the same object and the ***AO*** would be calculated in a uniform manner.

invert normal - this parameter inverts the direction of the ***AO*** calculation. If activated, the ***AO*** no longer applies dirt to the occluded zones but to the "freer" ones.

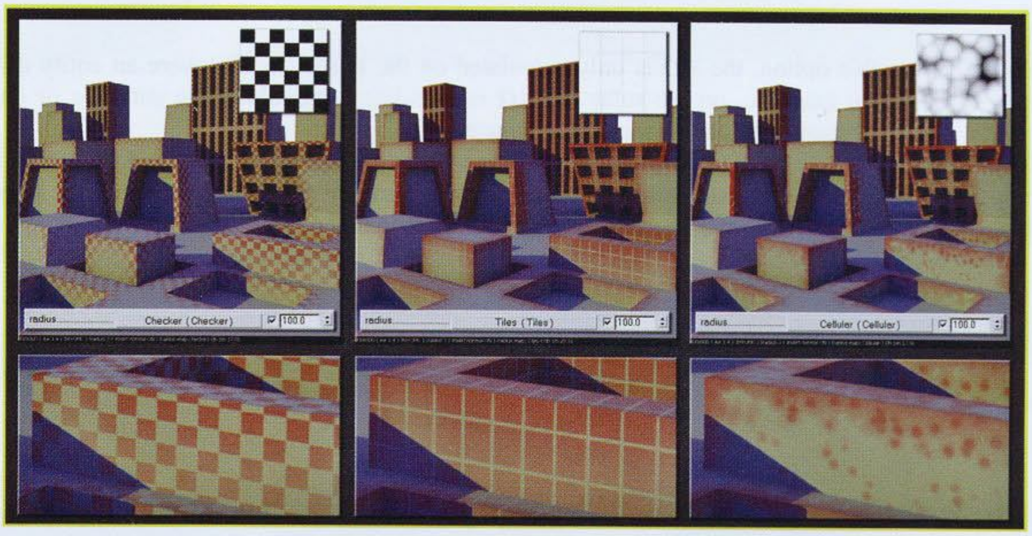
■ **Figure 5.265**
In the right-hand image the ***VRayDirt*** map behaves correctly, that is, it "dirties" the zones which are geometrically more occluded. By activating the ***invert normal*** parameter, the ***AO*** influences the more open zones. In the this model the areas in question are along the edges of the buildings.

radius map - by inserting a map into this channel, one can determine the radius of influence of the ***AO*** via a texture. The following image was created by using a ***VRayDirt*** map with the ***invert normal*** option activated and a ***radius*** value of 3.0.



■ **Figure 5.266**
Initial settings for an example of the ***radius map*** parameter.

So far there is nothing new about this. Now three maps will be used in the ***radius*** channel.

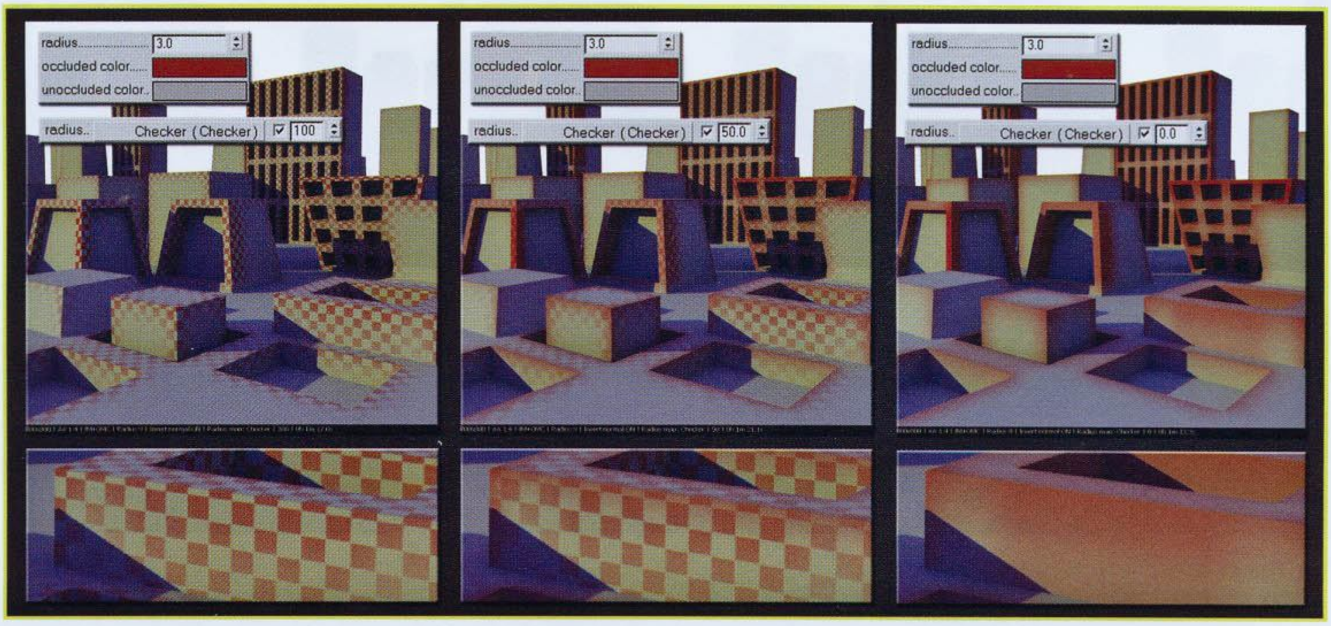


■ **Figure 5.267**
Procedural textures being used for defining the ***radius*** parameter.

In correspondence with the darker parts of textures, the radius of influence is none, there is no ***AO***. Vice versa, the radius is at its maximum in the white areas. Notice how a connection between the numeric value of the radius parameter and the textures being used exists and remains. For determining the radius of action of the ***AO***, in fact, one must specify it manually in the ***radius*** parameter.
In this case the value of the radius is 3.0 units. By changing the value and leaving the textures unchanged, ***AO*** influence increases due to the greater extension of the ***radius*** parameter.

■ **Figure 5.268**
Variation in the extension of the ***AO***.

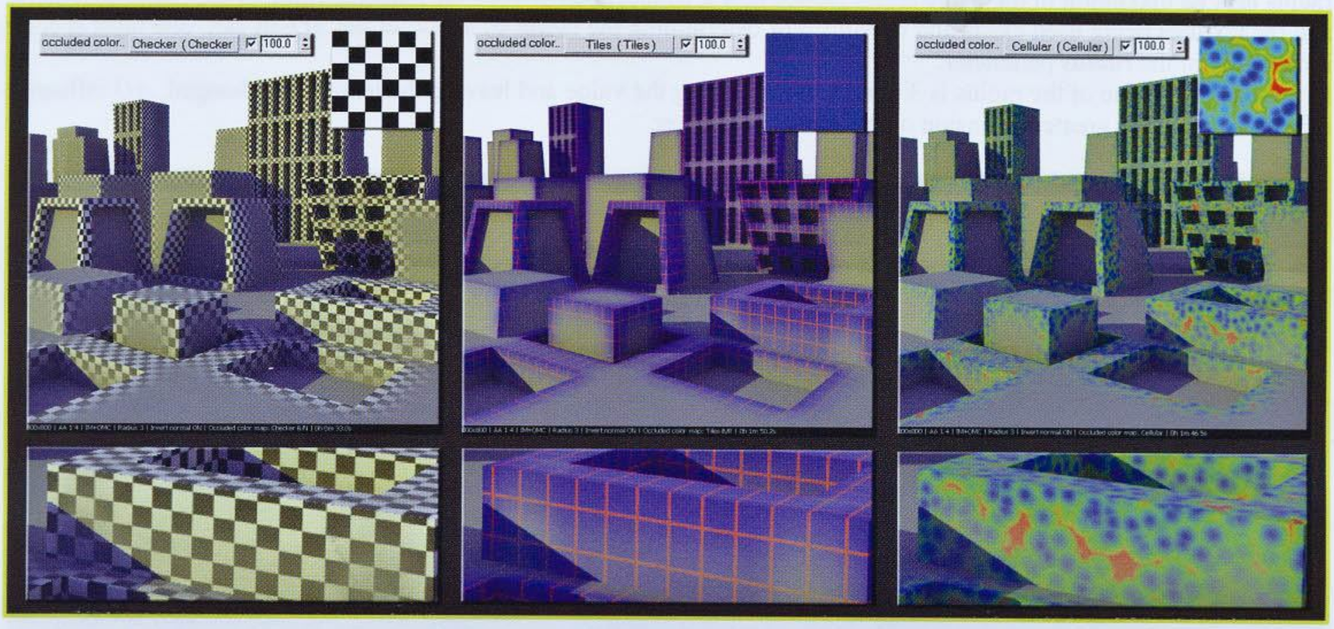
Lastly, the spinner dedicated to the ***radius*** channel which allows one to define the influence of textures on color. By lowering it one reduces the percentage of the use of the maps, therefore allowing a blend between textures and color.



■ **Figure 5.269**
Blending between texture and color.

As a last control option of the ***radius*** map, there is a small check to the left of the spinner. This allows textures to be disabled, and obtain the same result which one would have by resetting the spinner. With this button one can eliminate the influence of maps, without having to eliminate them physically or cancel their influence, which helps to speed up the setup of the scene.

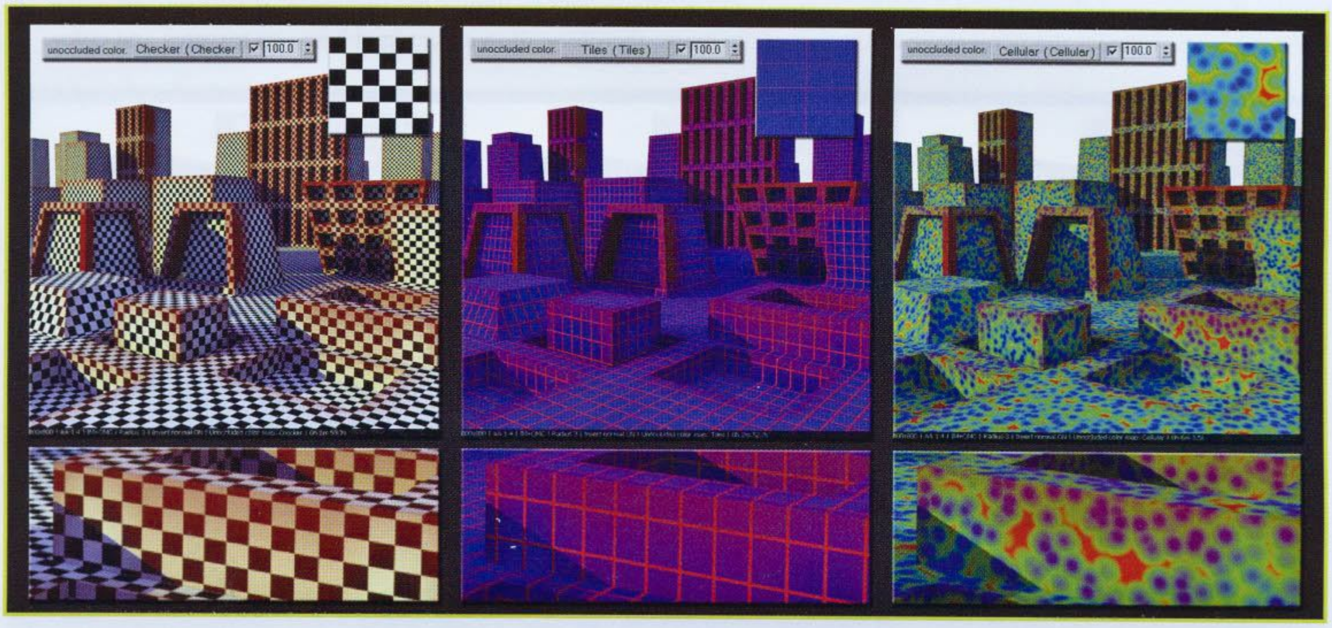
occluded color map - thanks to this channel one can decide the color of the ***AO*** by using a texture.



■ **Figure 5.270**
A map used for color occlusion.

In this case the map determines only the color of the ***AO***. With the spinner, as in the ***radius*** channel, one can mix the texture color with that of the ***occluded color*** map, instead with the button left of the spinner one deactivates ***occluded color*** map completely.

unoccluded color map - by inserting a texture in this map one can determine the inverse color of the ***AO***.

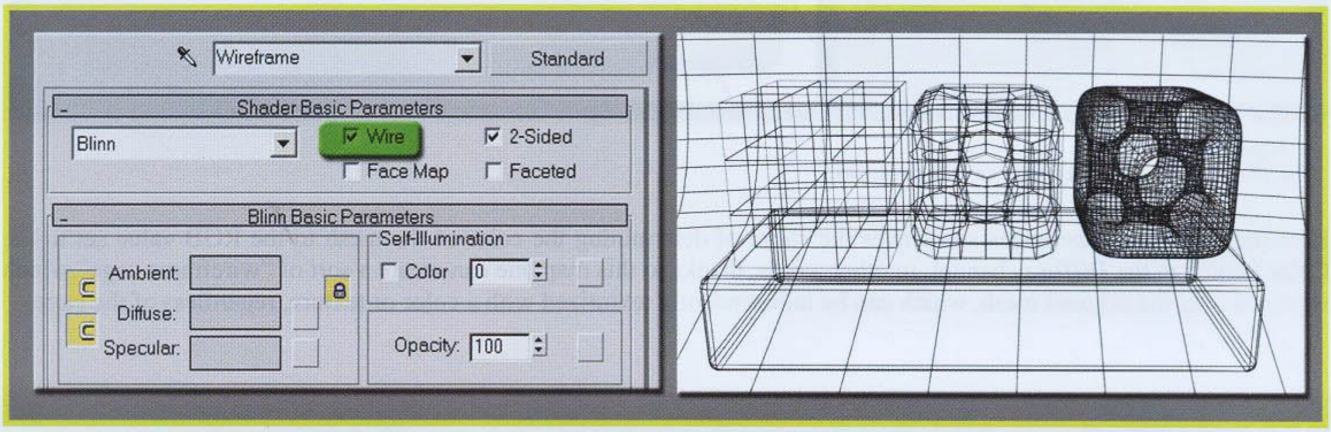


■ **Figure 5.271**
A map used for the inverse occluded color.

VRay Map: VRayEdgesTex

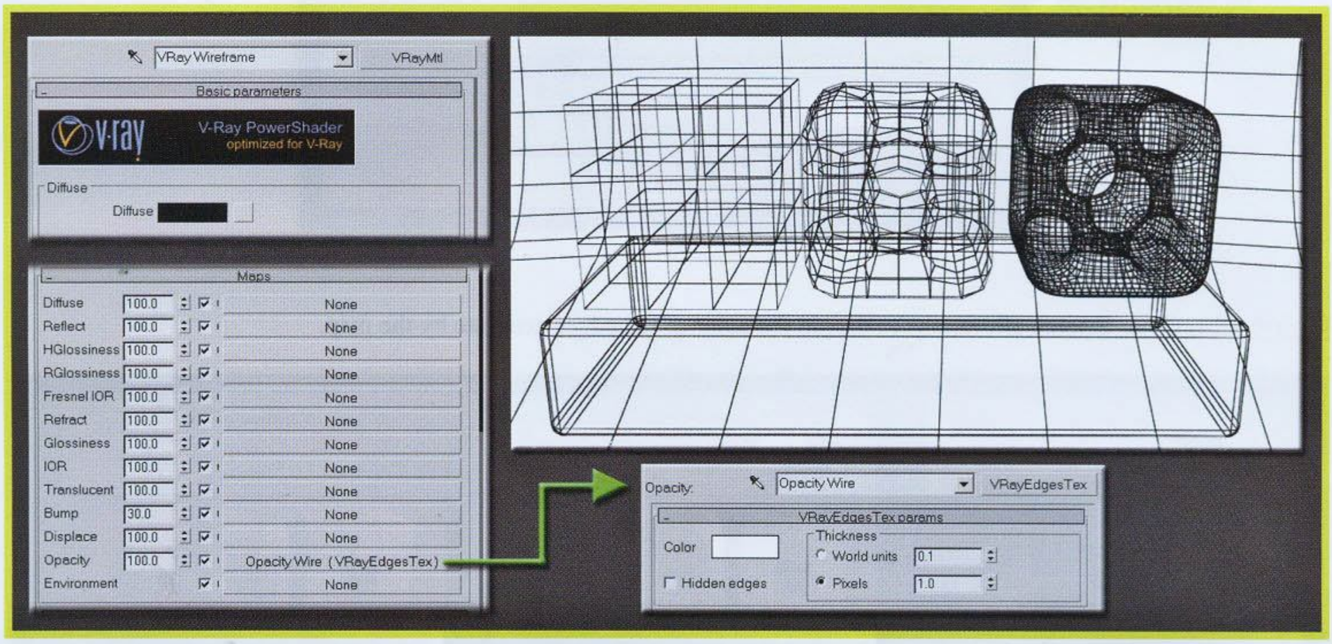
INTRODUCTION

One of the classic uses of ***VRayEdgeTex*** consists of simulating the "wire" effect, similar to the one generated via the activation of the Wire option in the Standard material of 3ds Max.



■ **Figure 5.272**
Render made through the Wire option in the Standard material. No light or particular effect has been applied.

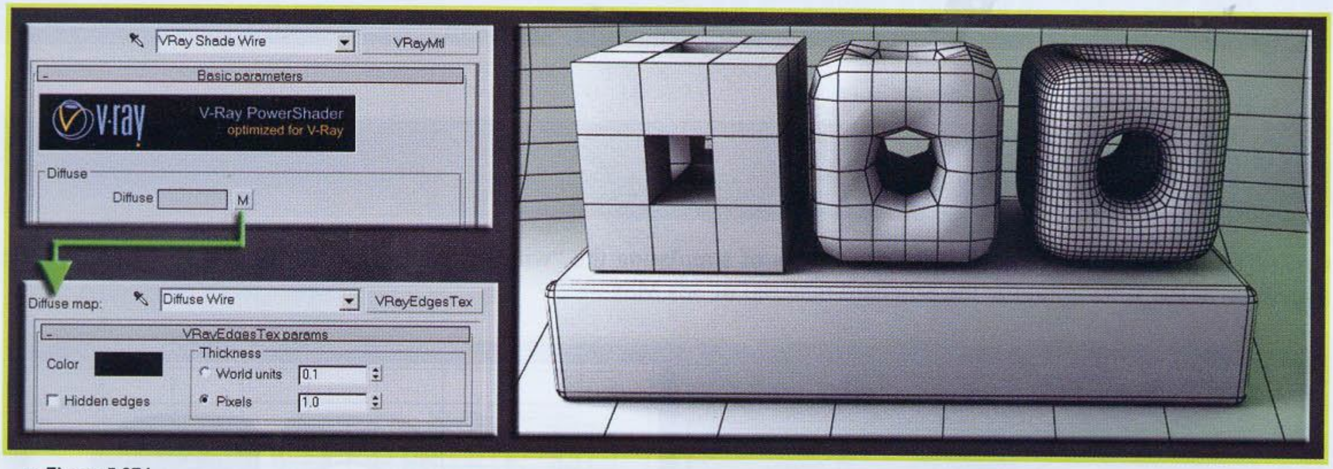
Unfortunately the Wire option of the 3ds Max Standard material is not compatible with ***VRay***. This and other reasons have lead to the creation of ***VRatEdgeTex***.



■ **Figure 5.273**
The same result obtained via VRay and its ***VRayEdgeTex***. Also here no light or GI effect has been used.

In this case the only measure that needs to be taken is to set Background to white, instead of the default black. In practical terms the ***VRayEdgeTex*** map is a texture which allows ***VRay*** to recognize edges of geometries and assign a color to them, which can be as thick as one wants. By inserting it in the opacity channel, the edges take on the color of the ***Diffuse*** channel, in our case black, instead the remaining parts become transparent. Also, the fact that it is a texture and not an effect like the 3ds Max Standard material means that it can be used in many occasions, granting better flexibility.

If one uses it as a map in the ***Diffuse*** channel one can obtain wireframe renders, but only in shade mode.



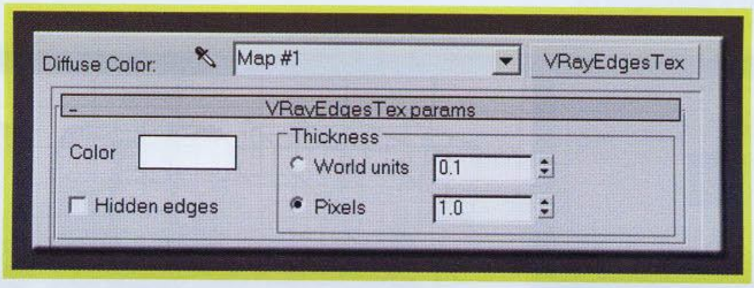
■ **Figure 5.274**
The ***VRayEdgeMap*** used as a map for the ***Diffuse*** channel.

Here the map colors the edges and leaves the chore of determining the color of the mesh to the RGB value set in the Color Switch in the ***Diffuse*** channel. In other words, thanks to this map one can create a sort of "wireframe cage" which is placed over the original mesh, which can be independently texturized with a color or texture, regardless of the cage.

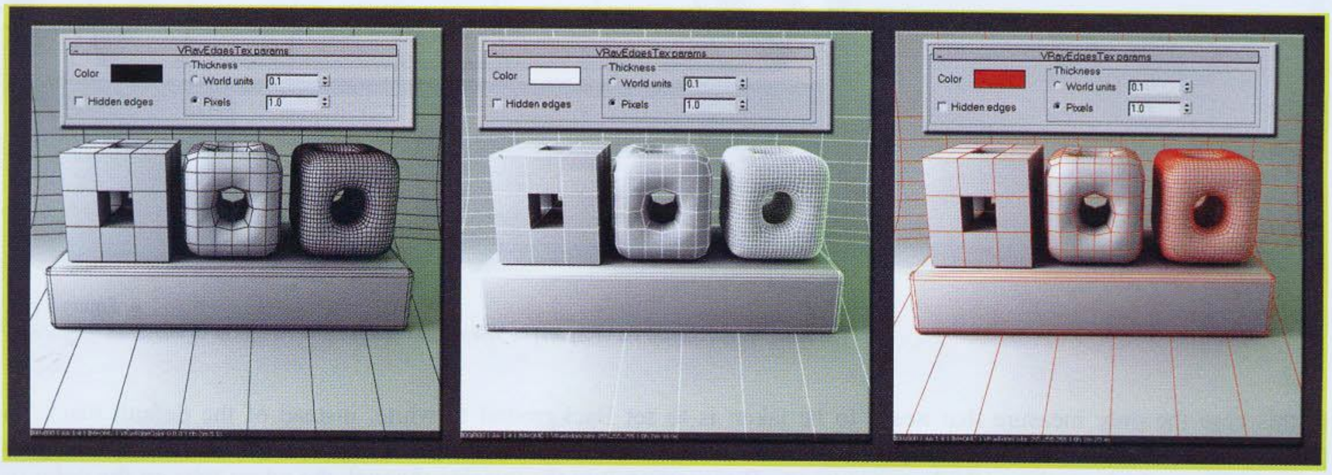
PARAMETERS

The ***VRayEdgeTex*** is a very simple parametric map with not many options.

■ **Figure 5.275**
The ***VRayEdgeTex*** map.

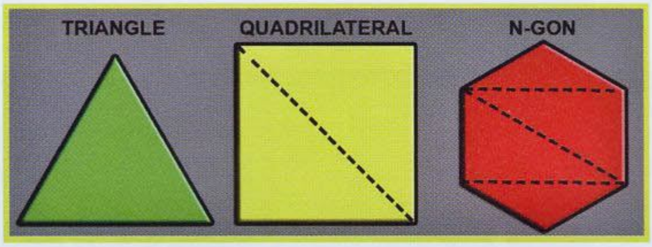


Color - this Color Swatch allows one to decide the color of the edges generate by the map.



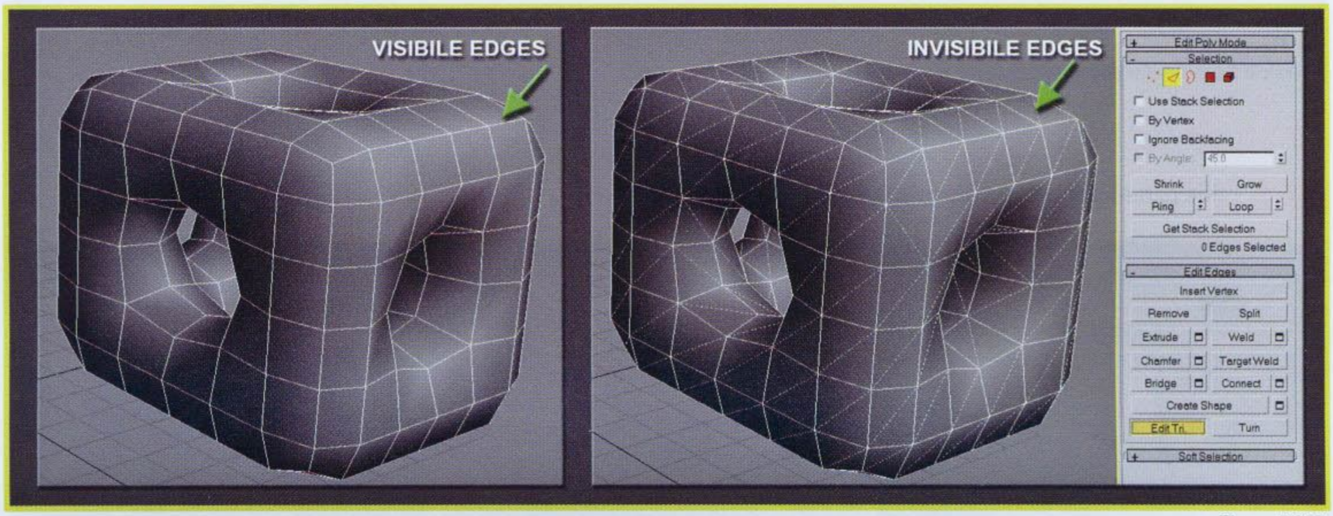
■ **Figure 5.276**
Different colors in the wireframe grid.

Hidden edges - all 3D software programs use triangles as basic building blocks. Although visually one can obtain polygons with more sides called n-gons, the 3D program elaborates geometries using triangles.



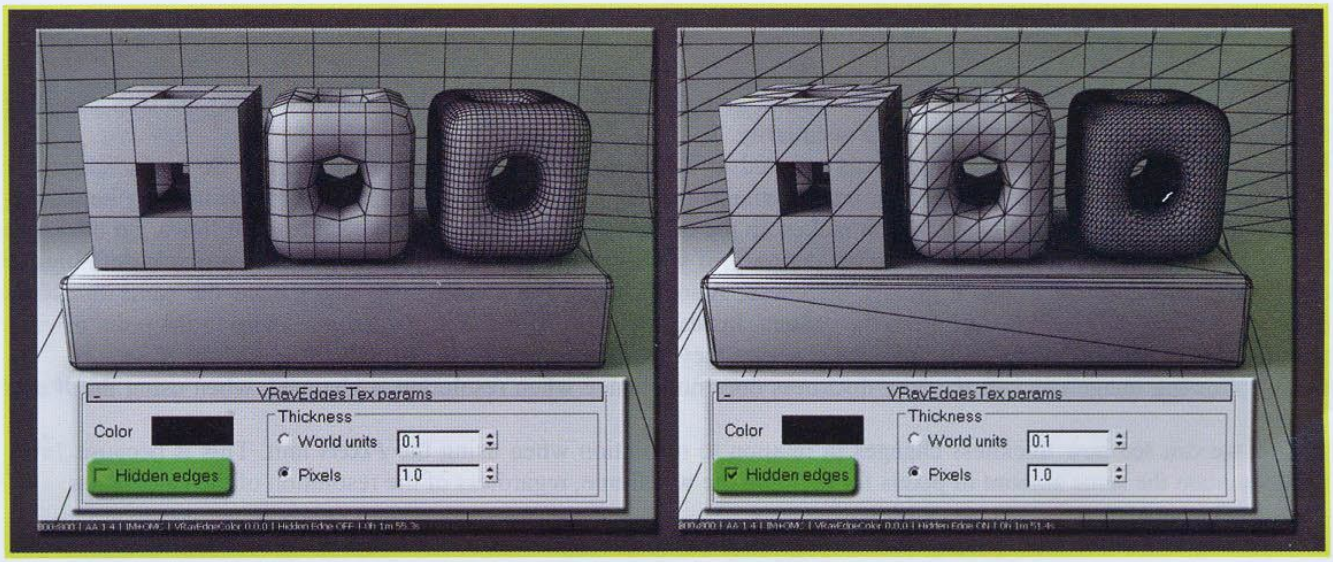
■ **Figure 5.277**
The basic figures in Computer Graphics.

3ds Max allows one to model via polygons with a big number of sides. One can observe the triangulation of the mesh anyway, by activating the Edit Tri option for Editable Poly.



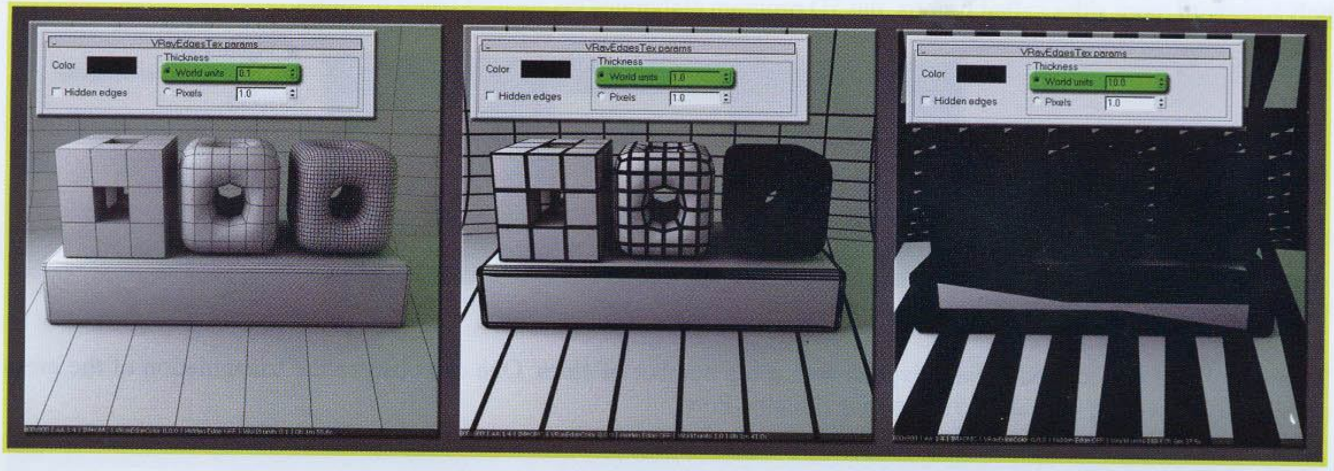
■ **Figure 5.278**
Visualization of hidden edges. These edges are represented with a dotted line.

By activating the ***Hidden edges*** option one can render hidden edges as well as visible ones.



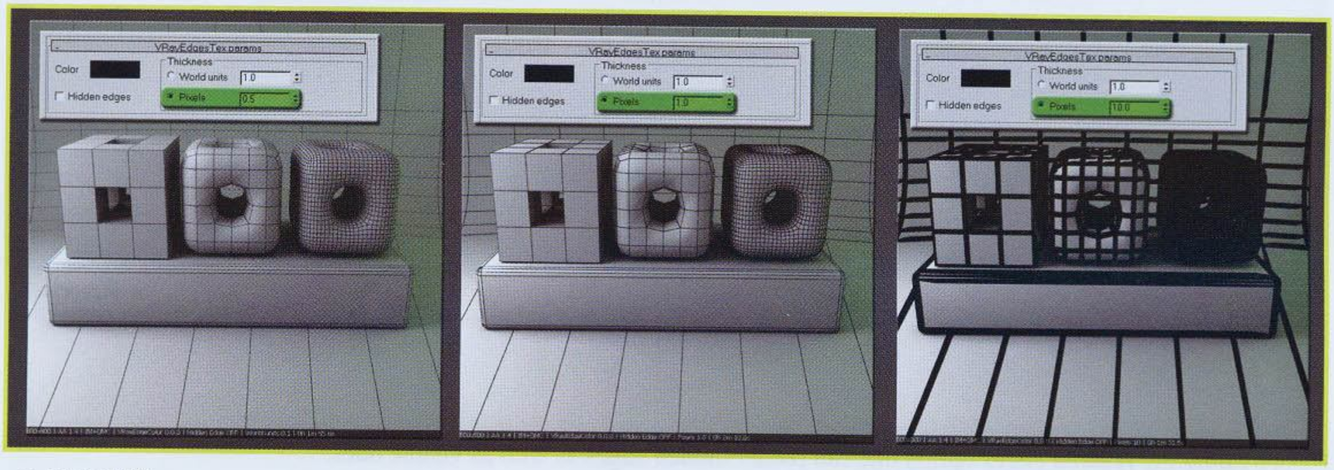
■ **Figure 5.279**
Rendering hidden edges.

Word units - this parameter is for deciding how thick edges should be, using the measuring unit in the 3D file.



■ **Figure 5.280**
Variable thickness of edges in relation to the ***World*** unit.

Pixels - this parameter lets one decide thickness measured in ***Pixels***.

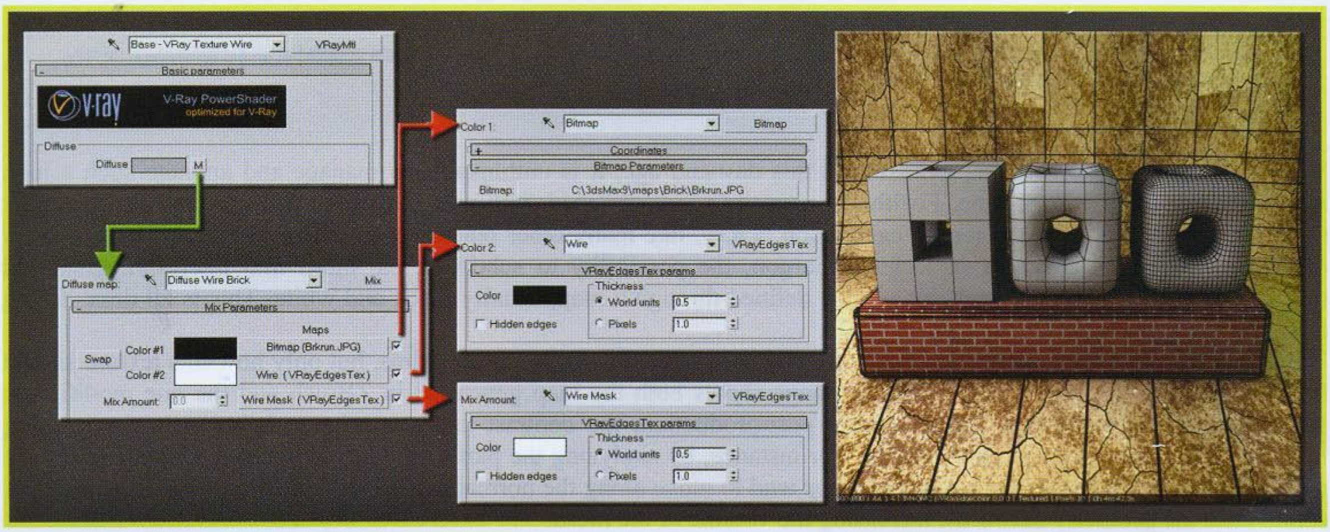


■ **Figure 5.281**
Variable thickness of edges in relation to the ***Pixel*** unit.

At a first glimpse it may seem as if these options give the same result, the only difference being the way the edge thickness is calculated. There is however a peculiarity.
In the first case the unit of measurement is referred to the model, in other words via real units. So by increasing or decreasing output resolution, thickness of edges remains unaltered. Vice versa, if we use the ***Pixels*** unit, by increasing the resolution, the number of pixels which makes up the thickness of edges must be the same at all times. So thickness decreases in relation to the geometry. On the contrary, if we lower the resolution, as the number of pixels defining the edges is constant, the grid has a greater thickness. Also by using the ***Pixels*** unit, edges have the same thickness regardless of how far way they are from the camera. Instead with the ***World*** unit, if edges are close to the point of view they are thicker. When further away from the observer they are thinner.
In the following image one can see how thickness does not change when resolution is changed, when using the ***World*** unit.
Instead we can see how thickness changes in relation to resolution when using the ***Pixels*** unit. This is because *VRay* must maintain the same amount of pixels for defining the wireframe, regardless of the resolution of the calculation.

■ Figure 5.282
Demonstration of the variation of edge thickness when varying the rendering resolution.

So far the ***VRayEdgeTex*** has been used for a wireframe effect, identical to that obtained with the Wire option in the Standard material. We have also seen how it can be used in shade mode, where the frame is placed over the color set in the ***Diffuse*** channel. It is also possible to create materials and effects which are more complex. In order to do this we can make use of the Mix material in 3ds Max. The base material has been defined with a rock Bitmap which we can mix with the ***VRayEdgeTex1***. At this point we define the blending mode. With the Mix Amount spinner we can only define the mixing percentage, so 50% Wire or 50% texture. The aim of this exercise is instead to use 100% of each map. For this reason another ***VRayEdgeTex2*** has been used. Now it is possible to tell 3ds Max which parts of the model to render with the Bitmap and which to render with the ***VRayEdgeTex1***.

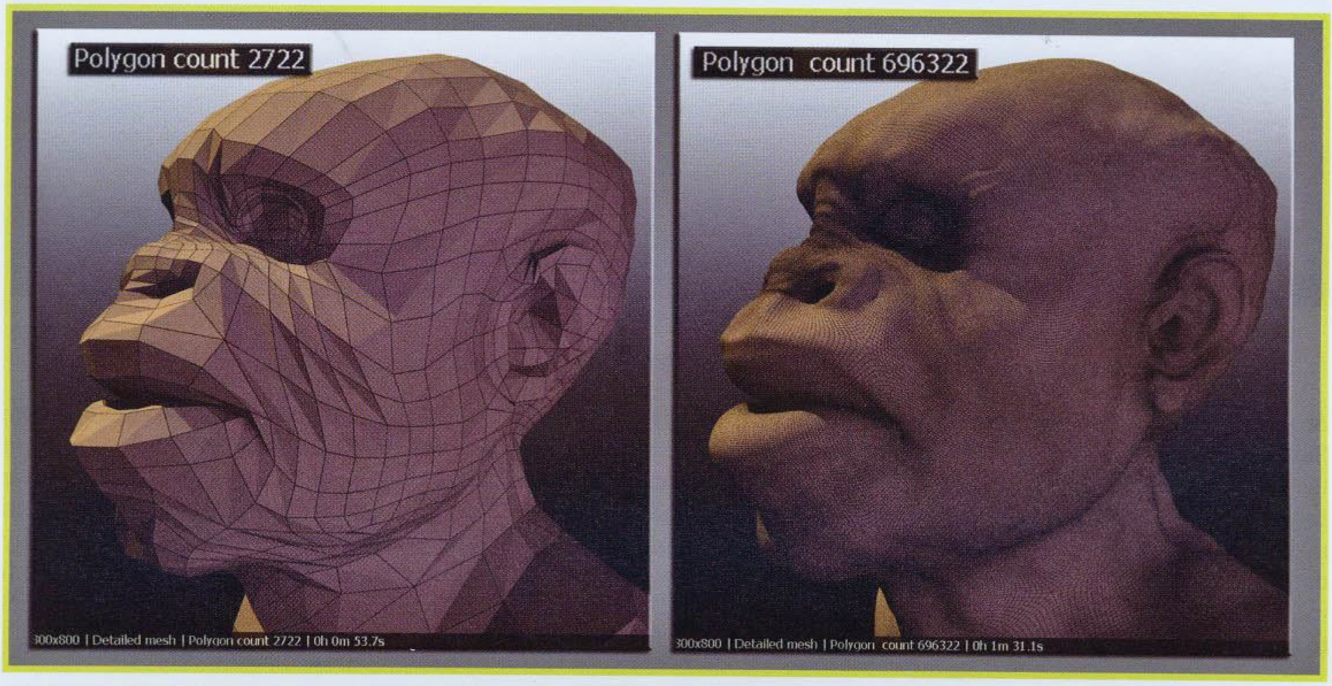


■ Figure 5.283
Simultaneous use of a Bitmap and the ***VRayEdgeTex*** map.

VRay Map: VRayBmpFilter

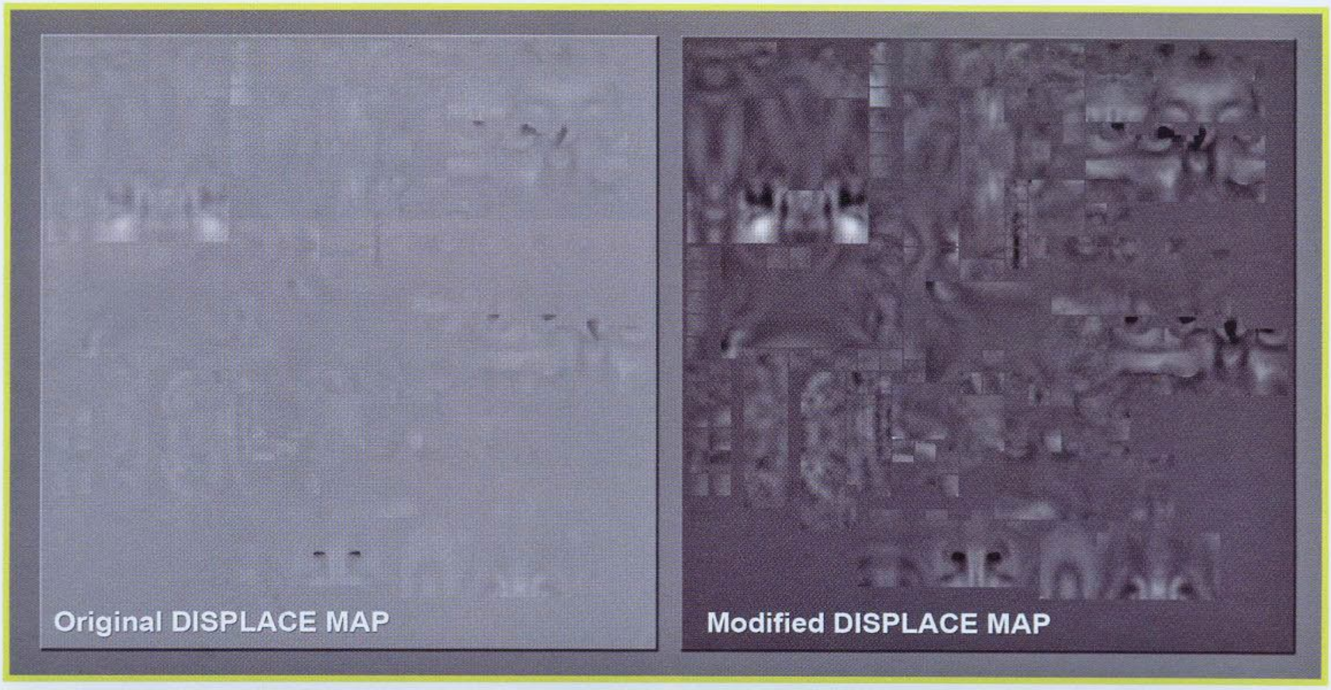
INTRODUCTION

In the last few years, thanks to ever more powerful software products, a particular type of modelling based on ***Displace*** maps has been spreading. ZBrush was the first software to spread this type of technology on the market. Lately other software programs are implementing a sculpting system within themselves which is similar to ZBrush. Its most direct competitor is called Mudbox, whereas other programs such as Silo or Blender have done nothing more than add ***Displace*** tools to the array of classic polygonal modelling tools.
Without wading too deep into the explanation of this subject, it can be summed up by saying that these programs allow one to sculpt a rough geometry, imported by other programs or created within them, and add extremely fine geometric details, easily reaching millions of polygons.



■ **Figure 5.284**
Mesh with different polygonal density. The image on the left is the starting model, the right-hand one shows the model sculpted at an extremely high density. The model was kindly provided by www.pixologic.com

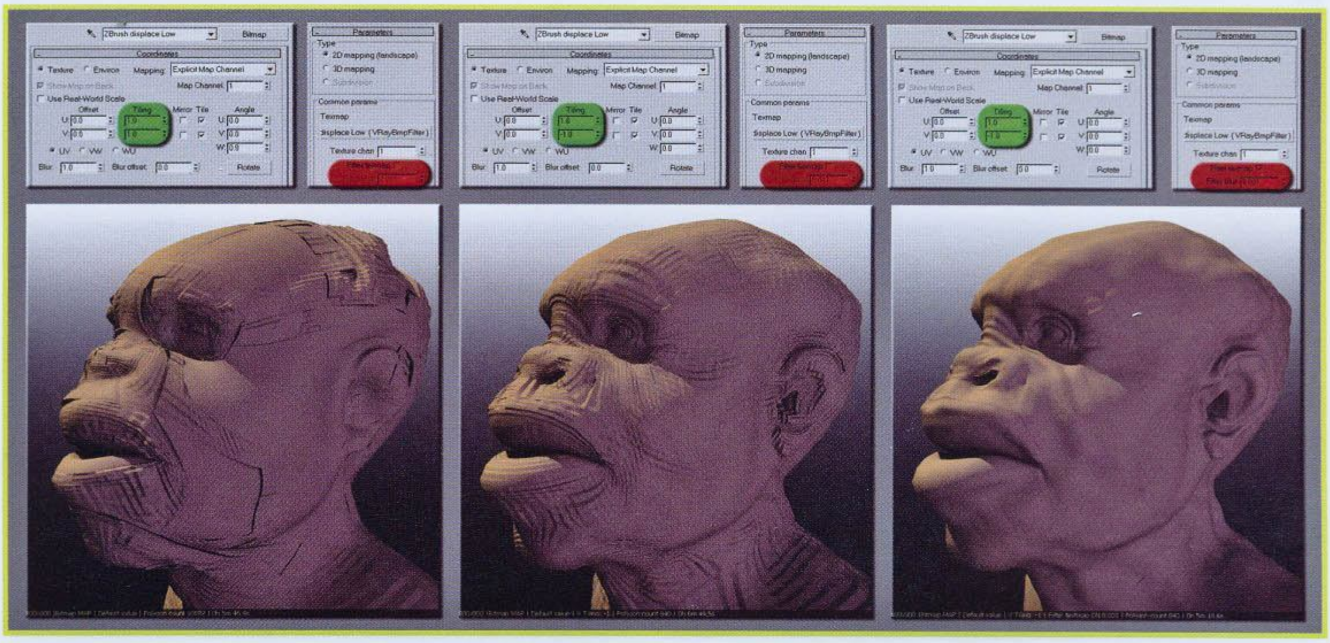
With the hardware and software available in this day and age, it is unthinkable to manage and animate such a high number of polygons in real-time. For this reason the result obtained with millions of triangles is created "at render time", that is, only during the render calculation. *VRay*, like many other rendering engines, is endowed with what is defined as micro-displacement in technical jargon, the possibility to subdivide the mesh into micro-polygons and translate them along their normals by a value defined through a map, usually black and white. This is all done during the render.
We will not explain the method for generating ***Displace*** maps for sculpting programs, but only how this texture should be treated within 3ds Max. The following image shows both the map generated in ZBrush and the one modified in Photoshop. In the right-hand image, during post-production, the contrast has been changed, but only for learning purposes. This makes details more visible. For instance, by analyzing the part of the map dedicated to the nostrils, the darker the map is, the more the micro-polygons tend to move along the normal negatively, retreating inwardly. Brighter colors push the geometry outwards, increasing the volume of the mesh.



■ **Figure 5.285**
Texture used for Displacement. On the right the same texture contrasted in Photoshop. This map was kindly provided by www.pixologic.com

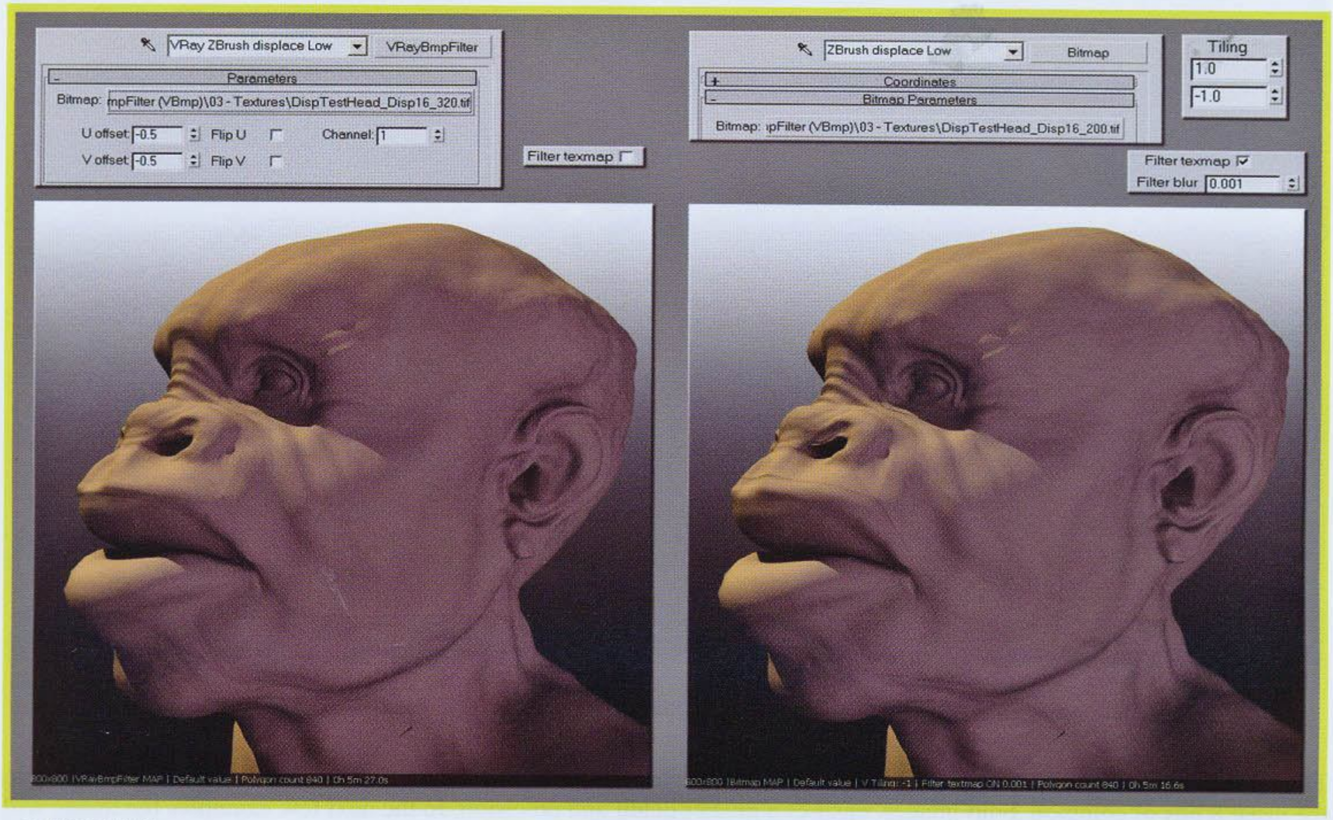
The map resolution is extremely high and in this case reaches 2048x2048. It is generated at 16 bit, which compared to the usual 8 bits of common .bmp images, ensures a greater detail and precision. In the next few examples the image will be reduced and brought down to 320x320 pixels. This will help us analyze ***VRay***'s behavior better. More precisely we will study the ***VRayBmpFilter*** and its management at pixel level of the map for ***Displace***, by comparing it with a classic Bitmap provided by 3ds Max.

First of all one must setup the scene. A Turbosmooth modifier is applied to increase the number of polygons on the model being examined, an anthropomorphic head. After a ***VRayDisplacementMod*** is added. In 2D mode a Bitmap is used with default settings. The result is rather strange. The texture is not correctly positioned and there are stair-effects due to the pixels of the ***Displacement*** texture. In order to solve the problem of the wrong ***Displacement*** map position, one must set the V Tiling, changing it from 1 to -1. This phenomenon is caused simply by a conflict between the UVW export settings between ZBrush and 3ds Max. The result is now better. The last problem is the stair-effect on single pixels due to the low ***Displacement*** map resolution. In order to solve this problem one must activate the ***Filter texmap*** option, and set it to the lowest value. Now the model is correctly rendered without flaws of this type.



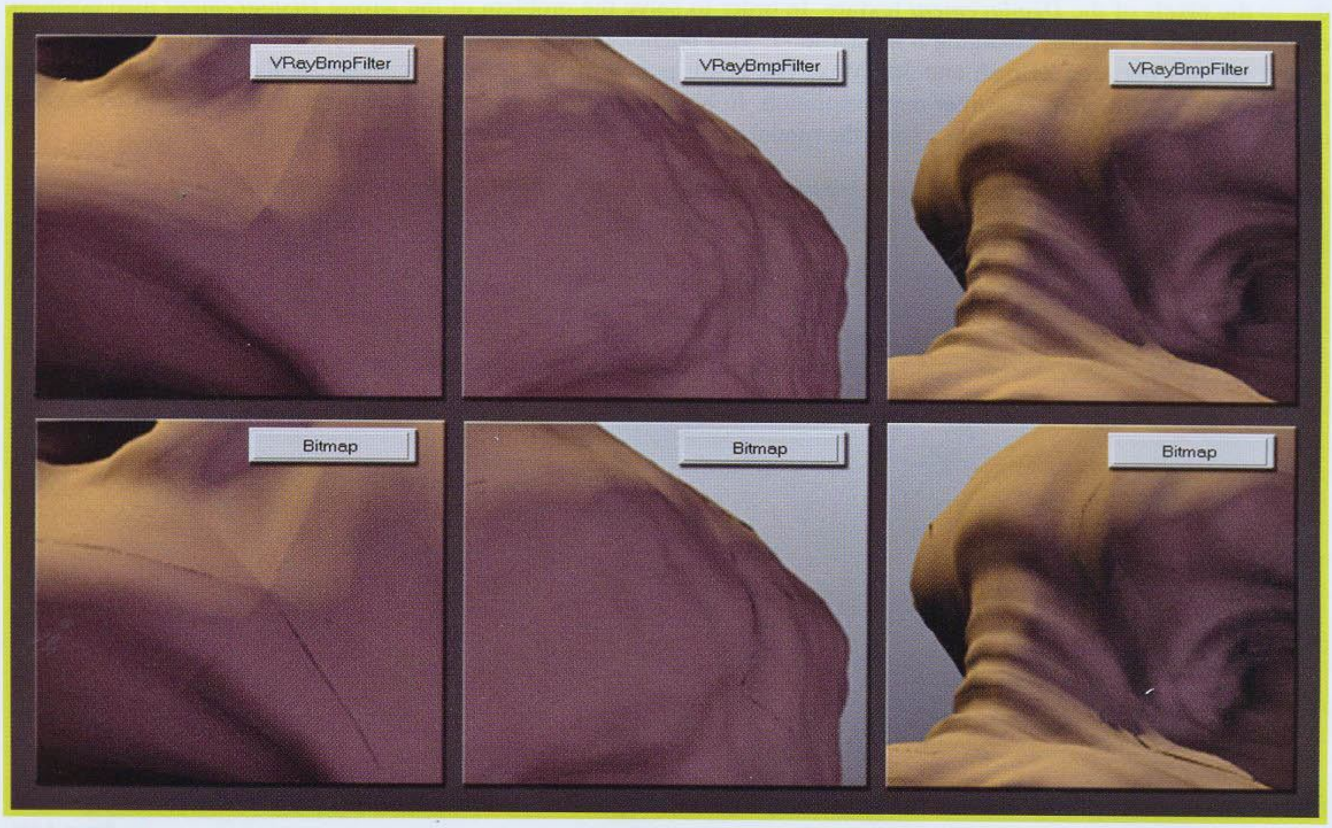
■ **Figure 5.286**
Use of a ***Displacement*** map created with ZBrush by using the 3ds Max Bitmap.

Now instead of the Bitmap the ***VRayBmpFilter*** will be used.



■ **Figure 5.287**
Comparison between the ***VRayBmpFilter*** and the ***Bitmap***.

At first the two renders seem identical, but there are actually some small but relevant differences. Firstly, in order to fulfill a correct render through the ***VRayBmpFilter***, there is no need to assign an orientation to the ***Displacement*** textures or to activate the ***Filter textmap*** option located in the ***VRayDisplacementMod*** modifier. Secondly, if we take a closer look at the two images, one can see at least two evident differences.

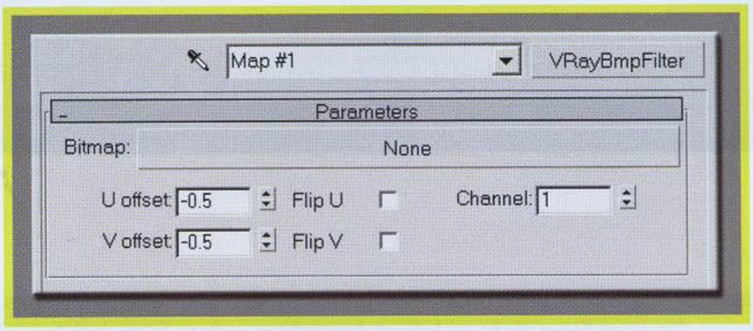


■ **Figure 5.288**
Image details.

The first difference concerns the edges of the conjunction between some parts of the map. When using the *VRayBmpFilter* the edges tend to disappear and are much more uniform than when using the simple Bitmap. The second difference is in the preservation of details. Remember how low the quality of the texture used for *Displace* is. And yet the *VRayBmpFilter* is able to maintain a certain level of detail, which is lost with the Bitmap.

PARAMETERS

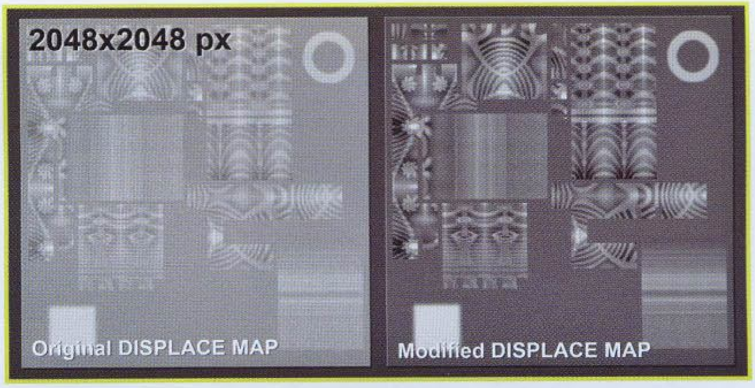
Similarly to the previous map, also in this case the parameters are few and easy to understand.



■ **Figure 5.289**
The *VRayBmpFilter* map.

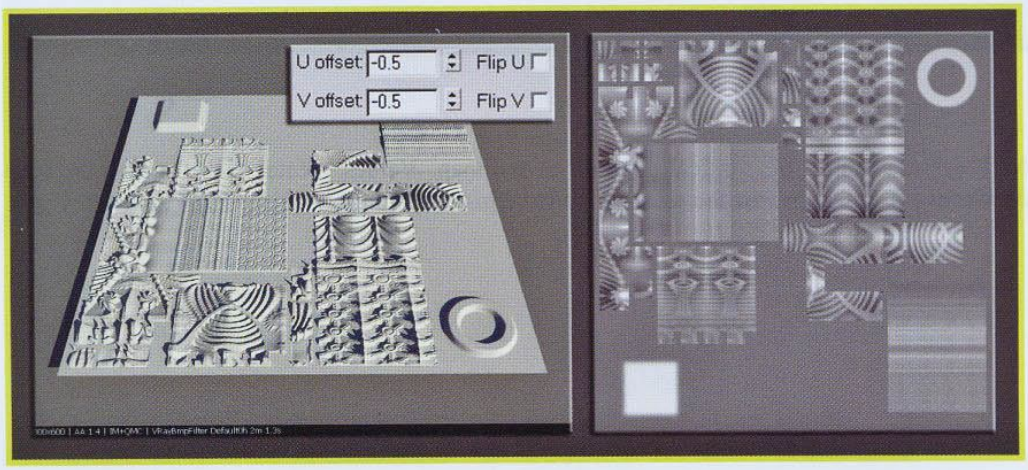
Bitmap - it is possible to insert any map or texture supported by 3ds Max. Here one can select its path.

U offset - by changing this parameter one can move the map along the U axis, with pixels for a unit of measure.
For the demonstration we will use another texture made in ZBrush. It is a *Displacement* map employed for texturizing a medieval object.



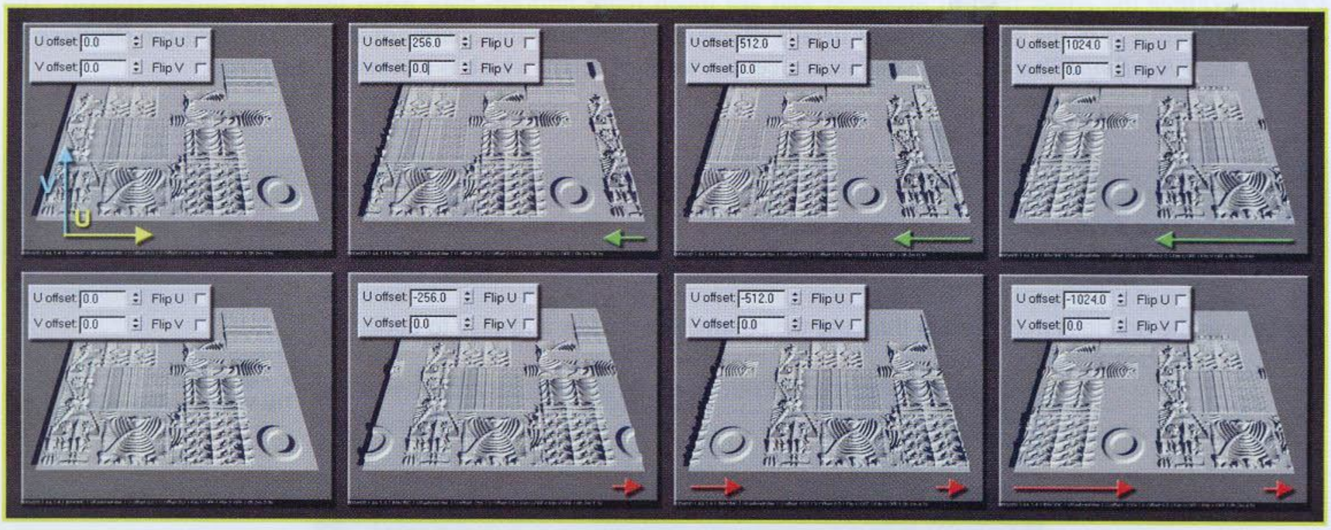
■ **Figure 5.290**
The map used for the next tests. The resolution is 2048x2048 pixels. This map is kindly provided by www.pixologic.com

In order to observe the change in parameters clearly, two images have been inserted in Photoshop: a circle and a square. With the *VRayBmpFilter* parameters set to default, this is the result.



■ **Figure 5.291**
A render fulfilled with the default *VRayBmpFilter* parameters.

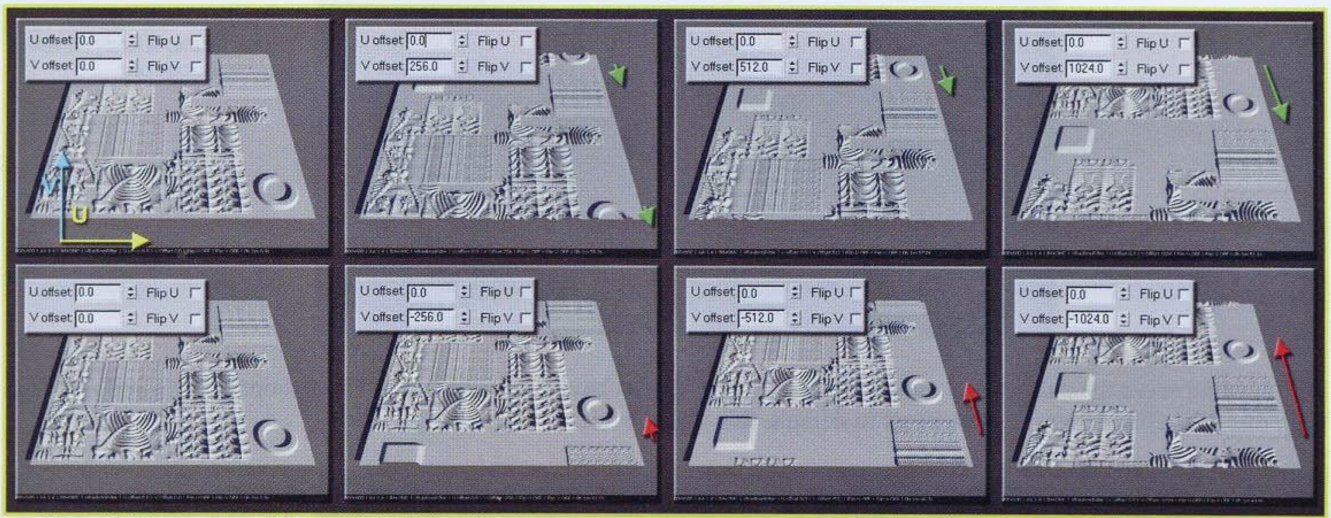
One can clearly see how in the render the map has been overturned along its horizontal axis..



■ **Figure 5.292**
U offset parameter changed.

Changing the ***U offset*** parameter implies a horizontal translation of the map and consequently of ***Displacement***. In particular, positive values move the map from right to left and negative values from left to right. The numbers inserted in this test are not randomly chosen. The map which has been used measures exactly 2048x2048 pixels. This means that as the ***U offset*** parameter uses pixels as its unit of measure, a value of 1024 is equivalent to exactly half a texture. With +1024 and -1024, in fact, the result is identical. The texture is moved first positively and then negatively by exactly half its size. The renders in this case are identical.

V offset - this parameter allows one to move the texture along the V axis, with pixels as a unit of measure.

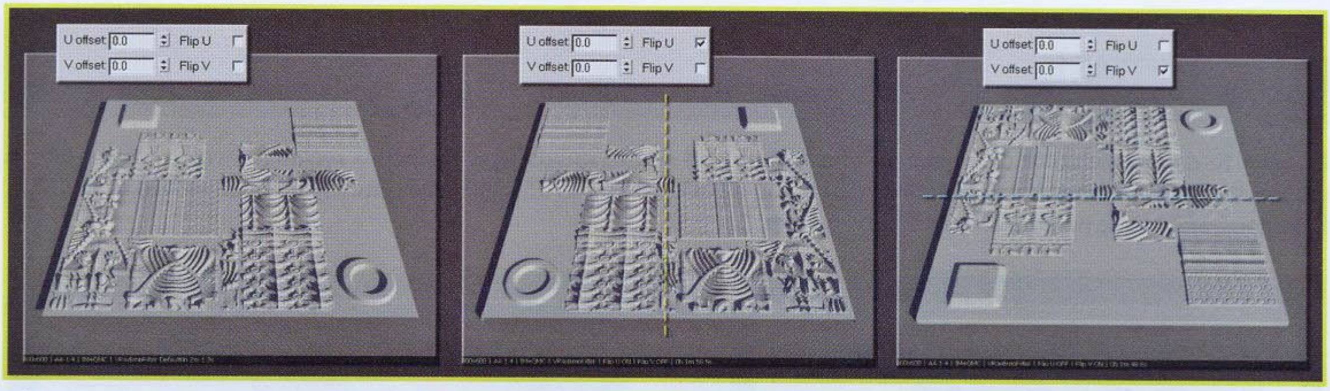


■ **Figure 5.293**
V offset parameter being changed.

Similarly to the ***U offset***, the ***V offset*** changes the vertical position of the map. Values greater than 0 move the map downwards and negative values move it upwards. For the reasons we have just explained, values +1024 and -1024 give us the same image.

Flip U - this parameter mirrors the texture along the U axis.

Flip V - this parameter mirrors the texture along the V axis.

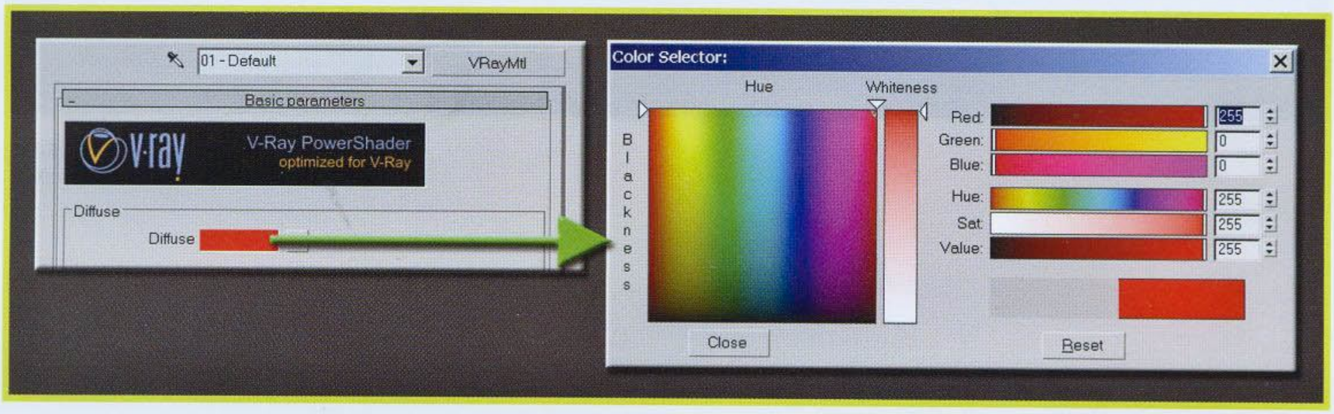


■ **Figure 5.294**
Texture flipped on the U and V axes.

VRay Map: VRayColor

INTRODUCTION

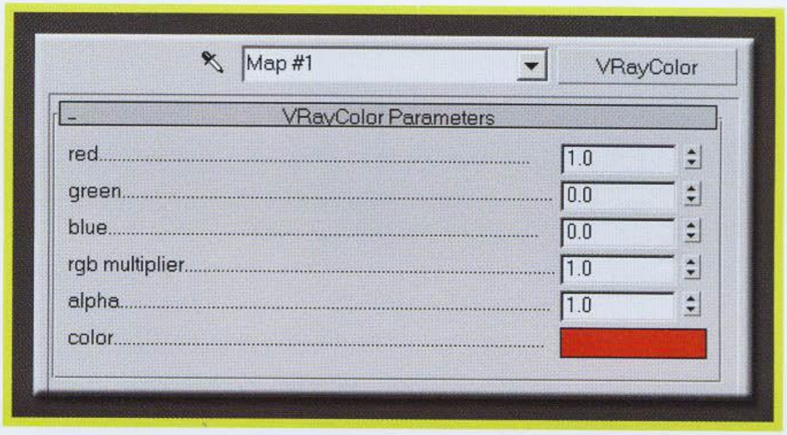
In order to define the color of a channel of a texture and so on, in 3ds Max there is a rectangle which allows one to modify the RGB parameters. This lets us define, on a scale from 0 to 255, the Red, Green and Blue values or the Hue, Saturation and Value parameters of any given color.



■ **Figure 5.295**
The 3ds Max Color Selector.

In theory it is possible to choose a color not only through the 256 classic variations of red, green and blue, but also via floating-point numbers which would allow us to obtain infinite colors. For a detailed explanation of this matter it is advisable to read the chapter about ***HDRI*** maps. Unfortunately, within 3ds Max, such colors cannot be defined (for instance a color such as RGB = [0.2-110.3-290.9]) because it does not recognize numbers below [1] and above [255] and which are fractionary. This has lead ***VRay*** to introduce its own color selector within any map: ***VRayColor***.

■ **Figure 5.296**
The *VRayColor* map.



PARAMETERS

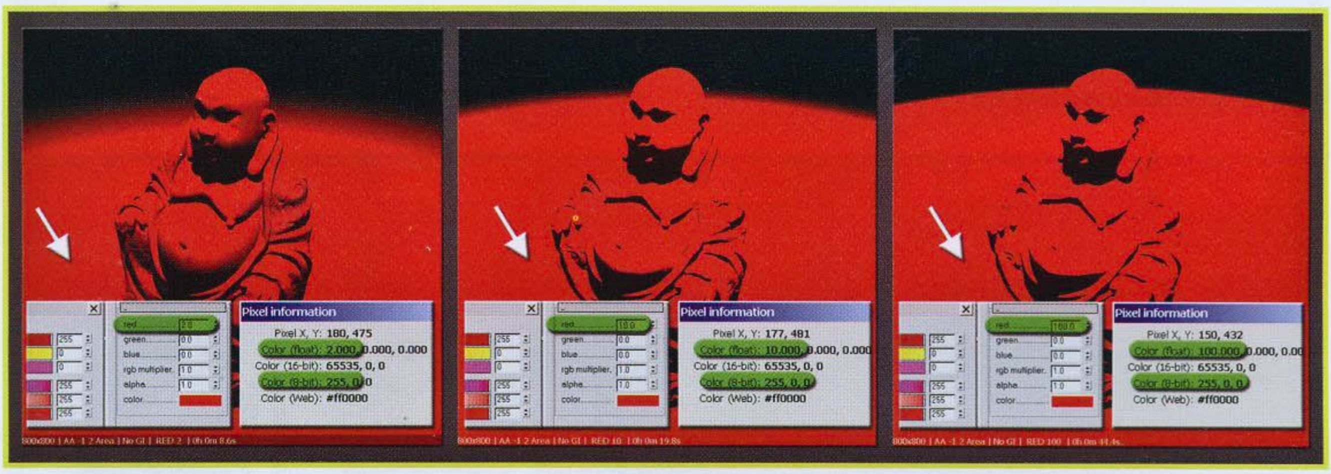
red - this parameter represents the value of the red channel. The value 0.0 in the classic RGB 0-255 scale corresponds to RGB = [0, 0, 0] whereas the value 1.0 corresponds to RGB =[255, 0, 0]. Value 0.5 corresponds to RGB = [128, 0, 0]. In order to represent the values in various channels of the classic RGB 0-255 scale, it is enough to use the following formula:

RGB value / 255 = VRayColor value

The scene examined is made up of a plane on which a statue rests, illuminated with a zenith unitary Direct light with no decay. The material is a ***VRayMtl*** with cyan ***Diffuse*** color, and the channel occupied by a ***VRayColor*** map with three different color tones: black, medium red and bright red. The scene has no GI.

■ **Figure 5.297**
Variation of the Red channel.

As one can see from the Color Selector, the ***VRayColor*** and from the ***Pixel information*** window of the ***VFB***, if one measures the illuminated plane, the result is a value which corresponds to the one in the ***VRayColor***. One can in any case use colors above RGB = [255,0,0]. If we do so, it is impossible to observe these colors on the monitor, because they are too bright for a classic CRT or LCD, although there are rare models on the market which are able to reproduce ***HDRI*** images.



■ **Figure 5.298**
Values above the usual 8 bits.

If we analyze the render with the ***Color Selector*** or the ***Pixel information*** window, the 8-bit depth color is always RGB= [255, 0, 0]. It would seem to be no different than the previous image. In actual fact, if we look at the floating-point value carefully, we can see that this is carefully considered and rendered correctly.

The following chart shows the conversion between some RGB scale values and the corresponding ***float number***.

RGB value		**VRayColor value**
[255,0,0]	....	[1.0 , 0 , 0]
[200,0,0]	....	[0.784 , 0 , 0]
[150,0,0]	....	[0.588 , 0 , 0]
[128,0,0]	....	[0.502 , 0 , 0]
[100,0,0]	....	[0.392 , 0 , 0]
[75,0,0]	....	[0.294 , 0 , 0]
[50,0,0]	....	[0.196 , 0 , 0]
[25,0,0]	....	[0.098 , 0 , 0]
[0,0,0]	....	[0 , 0 , 0]

green - this parameter represents the value of the green channel.

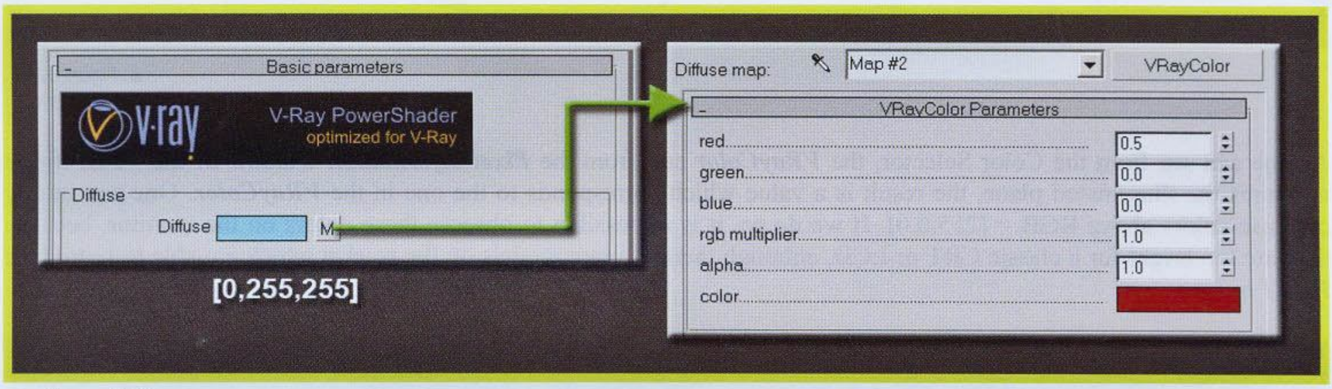
blue - this parameter represents the value of the blue channel.

rgb multiplier - this parameter is the multiplier for the RGB cannel. If we use a value of ***Red*** = 0.5 and a ***rgb multiplier*** value of =2, the color rendered will be 1.0, that is:

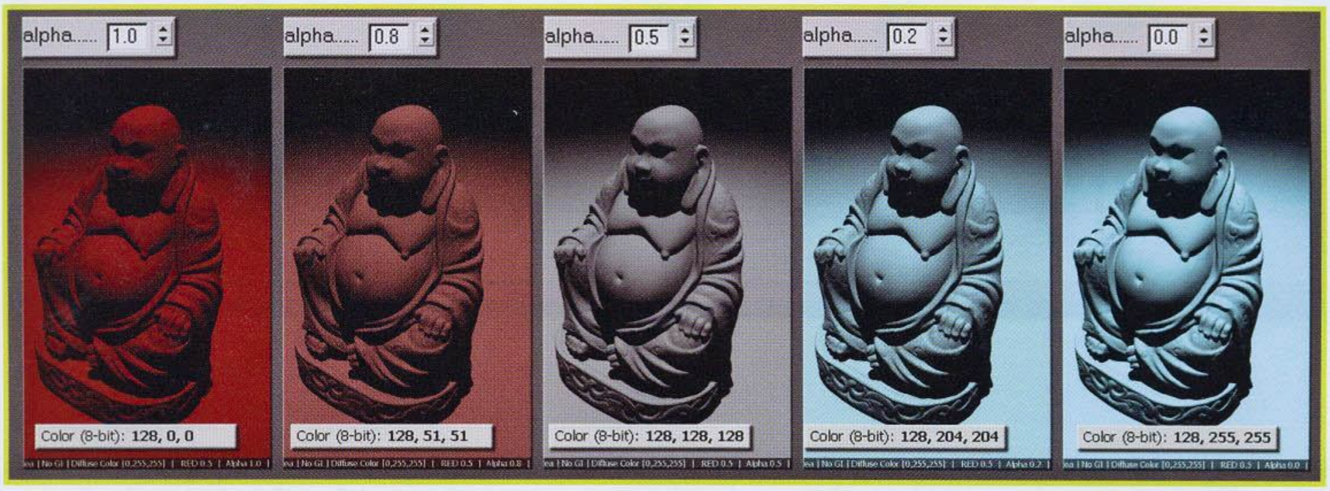
Final color = channel color x RGB multiplier

alpha - this parameter represents the value of the ***alpha*** channel. Value 0.0 corresponds to black and 1.0 to white. If for instance, this parameter is equivalent to 0.0, ***VRay*** uses the color of the material and mixes it with the RGB values of the ***VRayColor***. If it is set to 1.0, ***Diffuse*** color has no effect.

The material used in the tests is a ***Diffuse*** cyan, RGB = [0, 255, 255], with the channel occupied by the ***VRayColor***. In this case, ***VRayColor*** has only one channel, red = 0.5.



■ **Figure 5.299**
The material used for the ***VRayColor*** tests.



■ **Figure 5.300**
Use of different values of the ***alpha*** parameter.

Observe the functioning of the ***alpha*** parameter. The main color is defined by the ***VRayColor*** map. In this case it is an RGB color of RGB= [128, 0, 0]. This is because the red value is 0.5. The secondary color corresponds to cyan, RGB=[0, 255,255]. Why is the model grey with ***alpha*** = 0.5 ?

.With ***alpha*** = 1.0: color corresponds to 100% of the color of ***VRayColor***.

.With ***alpha*** = 0.8: the main color is mixed with 20% of the secondary color, that is, 20% of RGB = [128,51,51], which corresponds to RGB = [0,51,51]. Summed with the red value = 0.5 we get a RGB = [128,51,51] which is the color measured with the ***Color Selector*** at the base of the statue.

.With ***alpha*** = 0.5: the main color is mixed to 50% of the secondary color, that is 50% of RGB= [0,255,255] which corresponds to RGB = [0, 128, 128]. Summed to the value red = 0.5 we get an RGB = [128,128,128], an average grey.

.With ***alpha*** = 0.2: the main color is mixed with 80% of the secondary color, that is, 80% of RGB = [0, 255, 255] which corresponds to RGB = [0, 204, 204]. Summed with the value red = 0.5, we get a RGB = [128, 204, 204] which is a light blue tone.

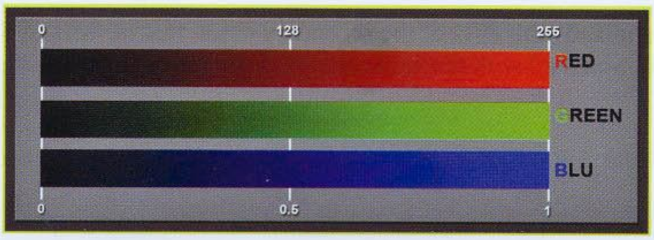
.With ***alpha*** = 0.0: the color of the ***VRayColor*** map is summed to 100% of the secondary color. We get an RGB=[128,255,255].

color - thanks to this box one can select the desired color interactively, and it will be reproduced in real time in the red, green and blue parameters.

VRay Map: VRayCompTex

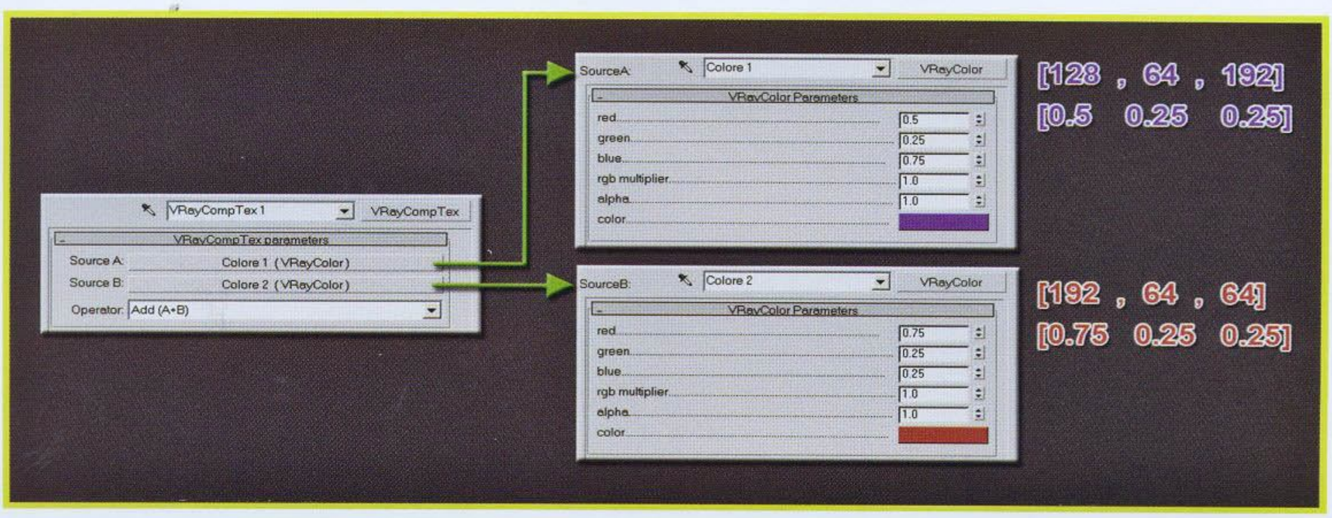
INTRODUCTION

Pixels are the smallest unit which make up a digital raster image. It can describe colors through different models and the most widespread is without an doubt RGB. This way, a color is represented by three main tones, identified in a scale which goes from 0 to 255, whose additive composition allows one to generate 16 million color combinations.



■ **Figure 5.301**
Three RGB colors and their relative values.

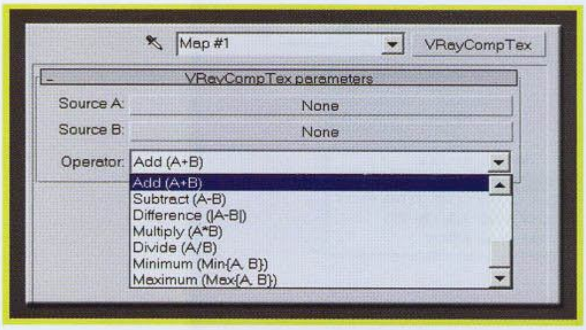
So red is identified by numbers with RGB = [255,0,0] or if it is normalized, [1,0,0]. One can combine colors, as they are represented with numbers, through the usual mathematical operations. They can be summed, subtracted, divided or multiplied.
The ***VRayCompTex*** does exactly this. It allows one to carry out mathematical operations with colors. The main purpose of this texture is during post-production. As the word suggests, the ***VRayCompTex*** is a texture (***tex***) of ***VRay*** (***VRay***) dedicated to Compositing (***Comp***). The term compositing stands for the discipline which takes care of all the procedures of assembling, modifying and correction which are applied to a render once its creation is prepared.
In most cases one fulfils a render in a single pass. Once the render is generated, from the ***VFB*** one can save the image in a user defined format. After this the image is ready to be printed. It is also possible to render the same image in a more complex way, by separating the various elements of the image on different levels, for them to be reunited later during post-production with programs such as Photoshop for static images or Combustion, After Effect or Nuke for animations. In this case, having various colored levels, one must then find a method for recomposing them, by summing, multiplying, dividing or subtracting them. This map can be useful if one does not have a post-production program available and one wishes to carry out this type of operation from within 3ds Max.
In the next few pages we will clarify the meaning of the options of sum, subtraction, difference, multiplication, division, maximum and minimum within the ***VRayCompTex*** map. To this purpose, two other source maps ***Source A*** and ***B*** will be associated with two ***VRayColor*** sub-materials. The ***VRayCompTex*** is used as a background map and therefore rendered.



■ **Figure 5.302**
The ***VRayCompTex*** map and two colors used for mathematical operations.

PARAMETERS

The *VRayCompTex* is a very simple map.



■ **Figure 5.303**
The ***VRayCompTex*** map.

In the two channels, ***Source A*** and ***Source B,*** one can insert any type of map, whereas with the pull-down menu one defines the type of mathematical operation to be applied to channels.

Source A - in this slot one can insert any map supported by 3ds Max. In the case being examined, we have used a ***VRayColor*** with RGB = [128, 64, 192], corresponding to a floating number value of RGB = [0.5, 0.25, 0.75].

Source B - in this slot one can insert any map supported by 3ds Max. In the case being examined, we have used a ***VRayColor*** with RGB = [192,64,64], corresponding to a floating number value of RGB = [0.75, 0.25, 0.25].

Operator - this allows one to choose, via a pull-down menu, one of the seven mathematical operations available in *VRay*.
In the following test, ***VRayCompTex*** is used as a background image in the scene. The ***VFB*** tool and ***Pixel information*** are used to analyze the final color, both in 8 bit terms and also ***float number*** value. In order to obtain the correct result, one must use float numbers as opposed to RGB values, because the latter are limited in the range which goes from 0 to 255, inexistent limits for the ***float number***.

- ADD - sums the values of the pixels of the two channels.
- SUBTRACT - subtracts the values of the pixels of the two channels.
- DIFFERENCE - sums the values of the pixels of the two channels and takes on its absolute value.
- MULTIPLY - multiplies the values of the pixels of the two channels.
- DIVIDE - divides RGB values.
- MINIMUM - takes on the minimum value.
- MAXIMUM - takes on the maximum value.

OPERATOR ADD

The operation ***Add*** sums the two operators.

RGB=[128,64,192]+
RGB=[192,64, 64]=

RGB=[320,128, 255]

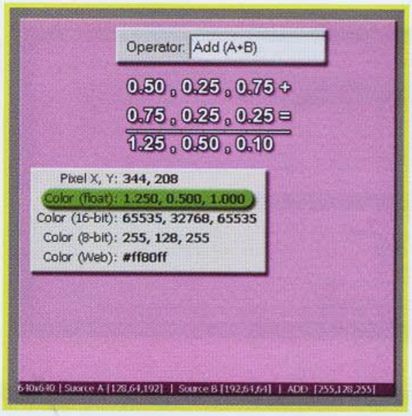
There is something wrong with this. The Red channel is equal to 320 and the maximum value in RGB can be at the most 255. If one observes the render in fact, the Color (8-bit) is RGB=[255, 128, 255]. The Red channel has been "clipped" to the maximum value available. In other words, important information on the Red channel has been lost. At this stage, if we convert the result to a ***float number***, we should get something like this:

RGB=[255,128,255] -> FLOAT=[1 , 0.5 , 1]

Instead the render carries the value:

FLOAT=[1.25 , 0.5 , 1]

This is the correct value because all the operations carried out with ***VRayCompTex***, as a demonstration of what we stated a few pages ago, use floating numbers as operators.



■ **Figure 5.304**
L'operazione ***Add (A+B)***.

OPERATOR SUBTRACT

The ***Subtract*** operation subtracts the two operators:

RGB=[128,64,192]-
RGB=[192,64, 64]=

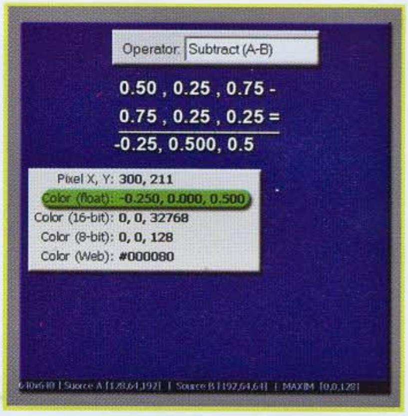
RGB=[-64, 0, 128]

This time the Red channel takes on a value which is not compatible with the RGB scale and therefore -64 is clipped to 0, the closest acceptable value in a scale of whole numbers [0,255].

By normalizing the RGB = [-64, 0, 128] number into a floating number, we obtain:

RGB=[-64, 0, 128] -> FLOAT=[-0.25 , 0.0 , 0.50]

as correctly shown by the render:



■ **Figure 5.305**
The Subtract operation ***(A-B)*** .

OPERATOR DIFFERENCE

The ***Difference*** operation subtracts the two operators:

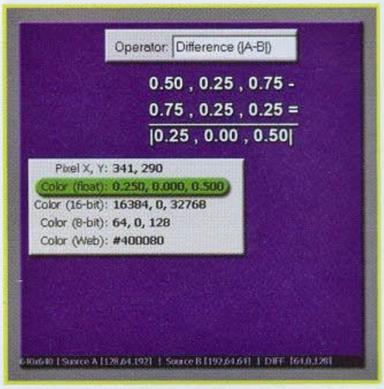
RGB=[128,64,192]-
RGB=[192,64, 64]=

RGB=[-64, 0, 128]

It then takes the absolute value, which is RGB = [64, 0 , 128] which in floating numbers is:

RGB=[64, 0, 128] -> FLOAT=[0.25 , 0.0 , 0.50]

as correctly shown by the render.



OPERATOR MULTIPLY

The ***Multiply*** operation multiplies the two operators:

RGB=[128,64,192]x
RGB=[192,64, 64]=

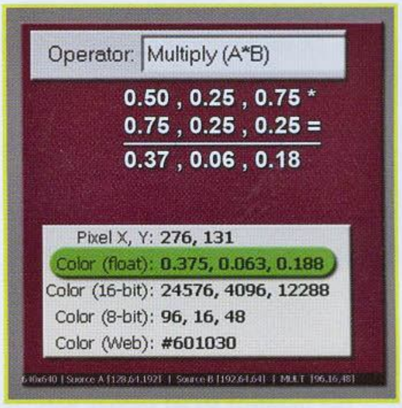
RGB=[24'575, 4'096, 12'288]

These are absolutely incorrect RGB values. With ***float numbers*** the result is:

FLOAT=[0.50 , 0.25 , 0.75]x
FLOAT=[0.75 , 0.25 , 0.25]=

FLOAT=[0.37 , 0.06 , 0.18]

as correctly shown by the render.



■ **Figure 5.307**
The Multiply operation ***(A*B)***.

If one transforms the final float number result in RGB, one gets RGB = [96, 16, 48], as in the final image.

OPERATOR DIVIDE

The ***Divide*** operation divides the two operators:

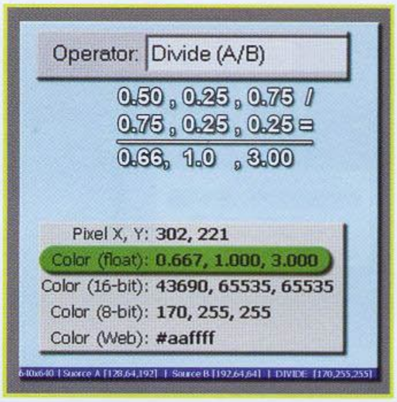
RGB=[128,64,192] /
RGB=[192,64, 64]=

RGB=[0.66, 1 , 3]

As in the previous examples, also in this case there are two channels with values which are not compatible with the RGB scale. Instead by using floating numbers, one obtains the correct result.

FLOAT=[0.50 , 0.25 , 0.75] /
FLOAT=[0.75 , 0.25 , 0.25]=

FLOAT=[0.66 , 1.0 , 3.0]



■ **Figure 5.308**
The ***Divide*** operation ***(A/B)***.

OPERATOR MINIMUM

The ***Minimum*** operation takes the smallest RGB values between the two maps.

RGB=[128,64,192]
RGB=[192,64, 64]

One obtains the following result:

RGB=[128,64,64] -> FLOAT=[0.50 , 0.25 , 0.25]

In this case, whether we use floating numbers or RGB values, the result remains the same.

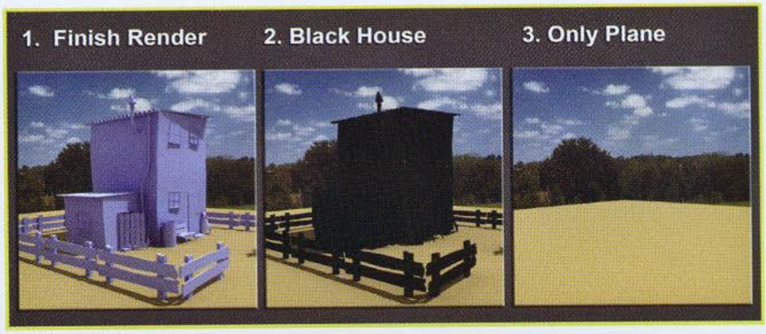
■ **Figure 5.309**
The ***Minimum*** operation ***(Min{A,B})***.

OPERATOR MAXIMUM

The ***Maximum*** operation takes the greatest RGB values between the two maps.

■ **Figure 5.310**
The ***Maximum*** operation ***(Max{A,B})***.

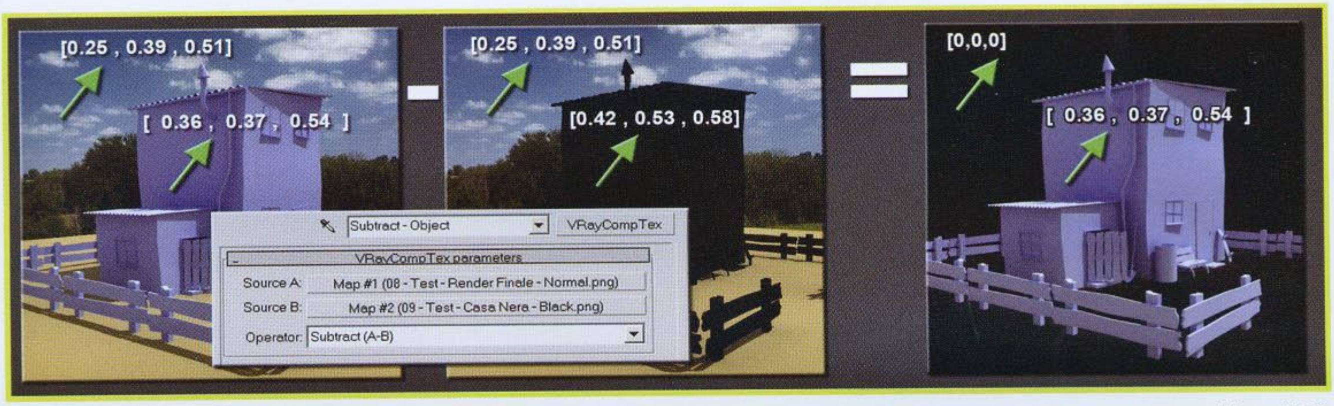
Without venturing into the world of compositing, some tests will now be shown which help to understand the ***VRayCompTex*** map.



■ **Figure 5.311**
The three images used in the test.

These three images were obtained through a separate render. The first is the most complex, the second instead was made by assigning a black colored ***VRayLightMtl*** to the building. In the last render only the horizontal plane was calculated.

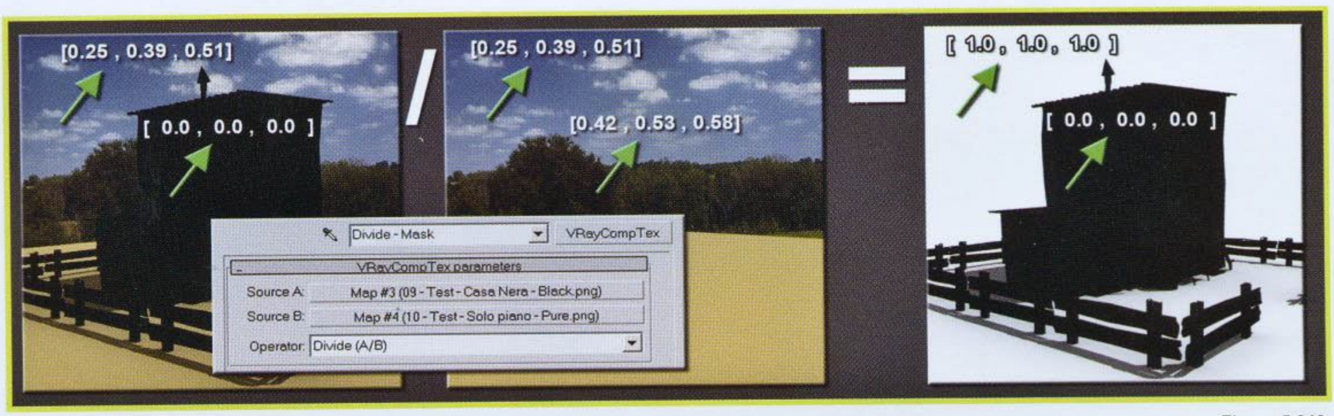
We will begin by using the first two renders elaborated via the ***Subtract*** operation.



■ **Figure 5.312**
Risultato dell'operazione ***Subtract***.

It is a simple operation and in this case also easily comprehensible. By subtracting the first image from the second, all the pixels of the first image are cancelled by the second image because they are identical. The only exception are the ones which are black. The result is that one isolates the object of interest: the building. For example, if we take a pixel from the first render which is equivalent to RGB = [65, 100, 130] and then subtract the same pixel from the second render, which is also RGB = [65, 100, 130], we are left with RGB = [0, 0, 0].

Now a division operation will be analyzed:



■ **Figure 5.313**
Result of the ***Divide*** operation.

The ***Divide*** operation, carried out with the renders 2. Black house and 3. Only plane, allows one to create a mask of the building. By dividing a number by itself, the result is always 1 and in the RGB scale this means RGB =[255, 255, 255], which is absolute white. Instead dividing RGB =[0, 0, 0] by any number gives us a final result of RGB = [0, 0, 0]. The final image is therefore made up of two colors only: black and white.

Without venturing into the world of compositing, some tests will now be shown which help to understand the *VRayCompTex* map.

■ Figure 8.311
The three images used in the tests

These three images were obtained through a separate render. The first is the most complex, the second instead was made by assigning a black colored *VRayLightMtl* to the building. In the last render only the horizontal plane was calculated.

We will begin by using the first two renders elaborated via the *Subtract* operation.

■ Figure [illegible]
[illegible]

It is a simple operation and in this case also easily comprehensible. By subtracting the first image from the second, all the pixels of the first image are cancelled by the second image because they are identical. The only exceptions are the ones which are black. The result is that one isolates the object of interest: the building. For example, if we take a pixel from the first render, which is equivalent to RGB = [68, 105, 158] and then subtract the same pixel from the second render, which is also RGB = [68, 105, 158] we are left with RGB = [0, 0, 0].

Now a division operation will be analyzed.

■ Figure [illegible]
Result of the Divide operation

The Divide operation, carried out with the renders 2. Black house and 3. Only plane, allows one to create a mask of the building. By dividing a number by itself, the result is always 1 and in the RGB scale this means RGB = [255, 255, 255] which is absolute white. Instead dividing RGB = [0, 0, 0] by any number gives us a final result of RGB = [0, 0, 0]. The final image is therefore made up of two colors only: black and white.

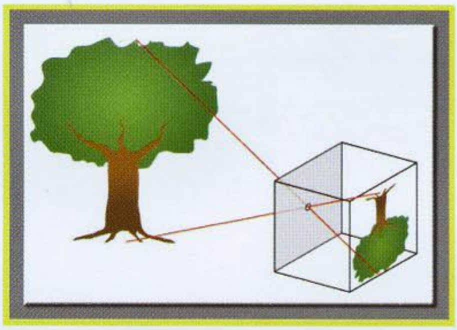
CHAPTER 6: VRAY CAMERAS AND LIGHTS

Theoretical concepts

THE CAMERA

One can think of the camera as a simple box in which a small hole is made. The light which goes through it is then projected upside down on the opposite side of the box. This primitive "dark room" is the basis of what will become the digital camera, a few hundred years later.

■ **Figure 6.1**
Schematic representation of a dark box.



Such a small hole could not let in much light and so in order to impress the image on the film placed within the box, it was necessary to expose it to the light for a long time. It was later discovered that a way to let more light in was to apply a lens to the hole, which permitted shorter exposure times and less problems concerning optical errors.

THE OBJECTIVE

The objective is one of the primary elements of the camera. It is an optical device placed on the front part of the camera and within it a number of lenses are contained. The main purpose of the objective is to gather the rays of light which make up the image and are to be projected on the film.

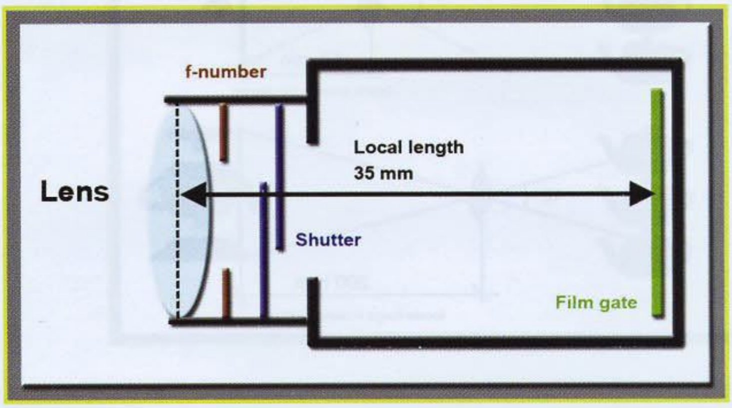
■ **Figure 6.2**
A camera without any objective attached and its family of optical extensions.

In actual fact, an objective could be made up of one lens only and in fact the first objectives were. Currently they are made up of a complex group of lens, of all different shapes and sizes, able to reproduce images with the highest level of fidelity possible. In Reflex cameras, as opposed to compact cameras, one can dismount the lens in favor of another one which has the characteristics. As well as the lens, the objective contains other components of fundamental importance, such as the aperture and the shutter, which as we will soon explain, allow one to decide how much light should enter and for how long the film or digital sensor are to be exposed to the light.

There are many types of objective, their main difference being their size. The parameter which defines their length is called **focal** and by using different objectives one can take photographs of subjects which are a long way away, landscapes or indoor scenes.

FOCAL LENGTH

The distance between the central lens in the objective and the focal plane (where the clear and focused image forms) is defined as the focal length of that lens. In the photographic field it is expressed in mm. In traditional photography, the format of reference is the most commonly used: the 35 mm.

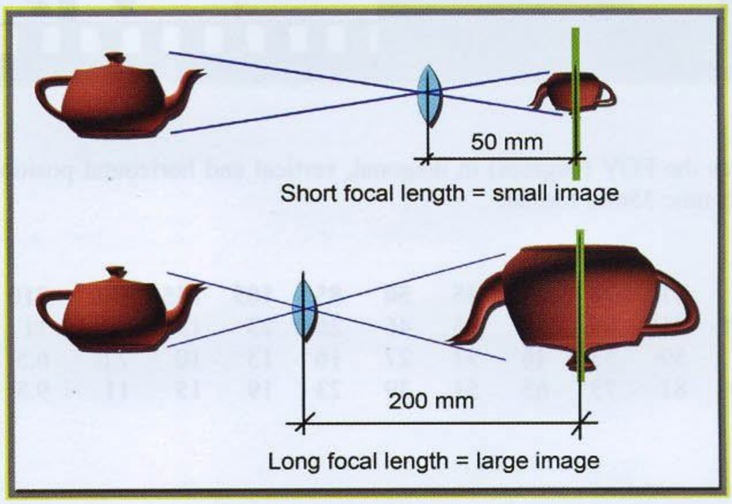


■ **Figure 6.3**
Scheme of a camera.

The objectives which cannot change their focal length are called fixed-focal objectives; if instead it can be changed, the objective takes on the name of variable focal objective or zoom. The focal length is indicated in fixed objectives with a number expressed in millimeters, for example 180mm, instead for zooms there are two values which define the range of possible focal lengths which can be obtained, for example 35 - 135 mm.

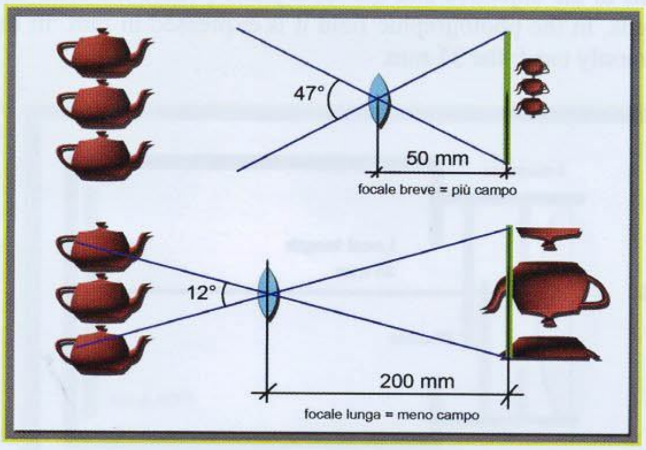
■ **Figure 6.4**
Two types of objective, the first one with variable focal length and the other fixed.

Observe the following image which demonstrates the difference between the two objectives.



■ **Figure 6.5**
Scheme showing how two types of objectives with different focal lengths work.

If one maintains the same object to be photographed, but changes the objectives only, going from a short to long focal length, one finds that the object appears larger on film, and takes up more space on the photosensitive surface. These particular objectives, also called teleobjectives, are used to capture very far away subjects. One can therefore state that the size of a subject photographed with a 50 mm objective is twice as large as its size when using a 25 mm and half as big as its size when using a 100 mm objective. Focal length, as well as influencing the relative enlargement of an image, also determines the portion of space which can be framed with an objective. This portion of space is defined by the field of view and is inversely proportional to the focal length of the objective.

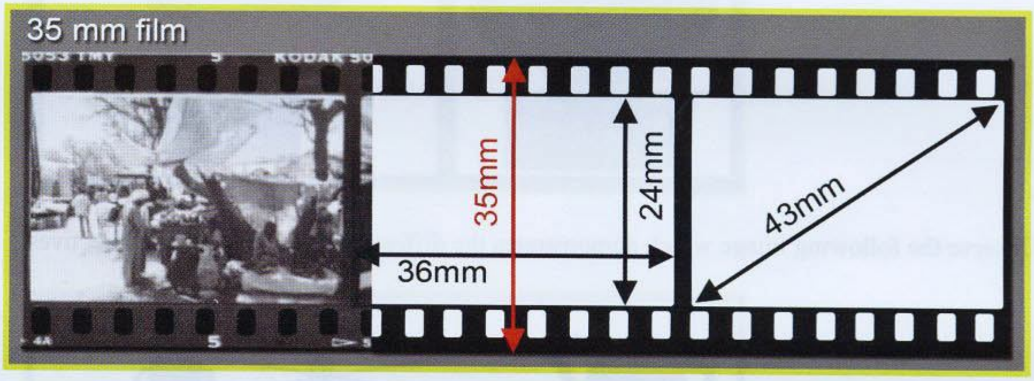


■ **Figure 6.6**
Modification of the angle of vision according to the focal length. Here the film size is 35mm, not to be confused with focal length which is also expressed in mm!

The greater the focal length is, the smaller the field of view which can be framed. For example, an objective of 200mm has a field angle of 12° (always considering a 35 mm film); vice versa, in order to photograph vast areas, one must use objective with a short focal length, which have a significant field of view. For instance a 50 mm has a field of view of 47°. The field of view is therefore determined by two variables: focal length and the size of the sensor or film. At equal focal length, in fact, if we change the size of the film, also the field angle changes. In the previous image, if one reduces the size of the film (the green vertical bar) and leaves the rest unaltered, the FOV.
Most of the major producers of objectives offer a vast range of focal lengths between 12mm to 600mm. The FOV usually indicates the diagonal angle of vision and not the horizontal or vertical one.
The formula for calculating the FOV is:

Field of View = 2*arctan(d/2f)

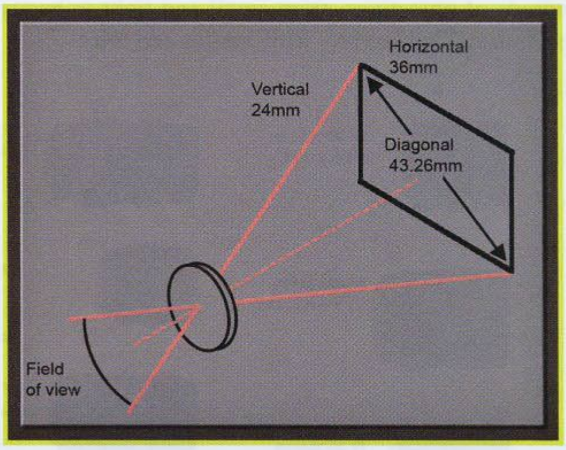
where **d** is the diagonal of the film, **f** is the focal length. There are various types of film but the most widespread for amateur use is 35mm, which has a diagonal measurement of 43.266mm.



■ **Figure 6.7**
A photographic film of 35mm. With a 50mm objective one has a field of view of around 46°.

The following chart indicates the FOV (degrees) in diagonal, vertical and horizontal positions given by various focal length settings (mm) in the classic 35mm format.

	13	**15**	**18**	**21**	**24**	**28**	**35**	**50**	**85**	**105**	**135**	**180**	**210**	**300**	**400**	**500**	**1200**
DIAG.	118	111	100	91	84	75	63	46	28	23	18	13	11	8.2	6.1	4.9	2.07
VERT.	85	77	67	59	53	46	37	27	16	13	10	7.6	6.5	4.5	3.4	2.7	1.1
HORIZ.	108	100	90	81	73	65	54	39	23	19	15	11	9.8	6.8	5.1	4.1	1.7



■ **Figure 6.8**
Schematic representation of the field of view.

From the previous formula it is easy to calculate the FOV of a 50mm objective used with a film of 35mm.

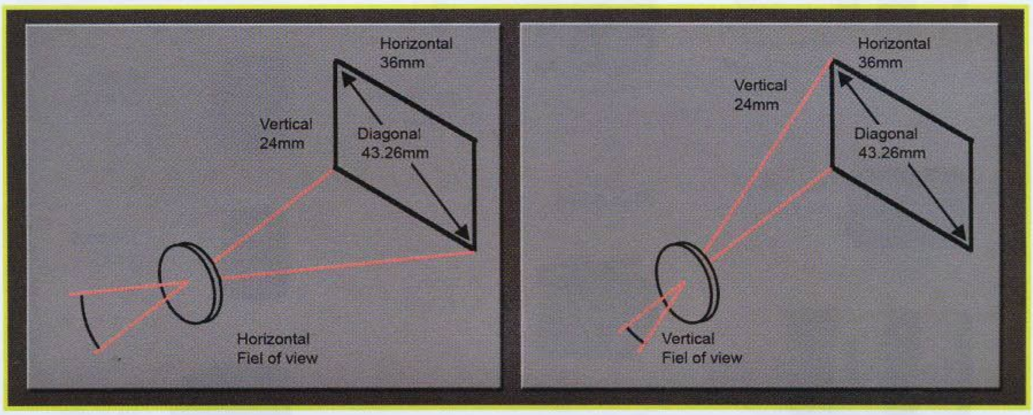
Field of view = 2*arctan(d/2f) -> 46.8° = 2*arctan[43.26/(50mm*2)]

In order to find the vertical and horizontal field of view, it is enough to use the following formulas:

Horizontal field of view = 2*arctan(o/2f)
Vertical field of view = 2*arctan(v/2f)

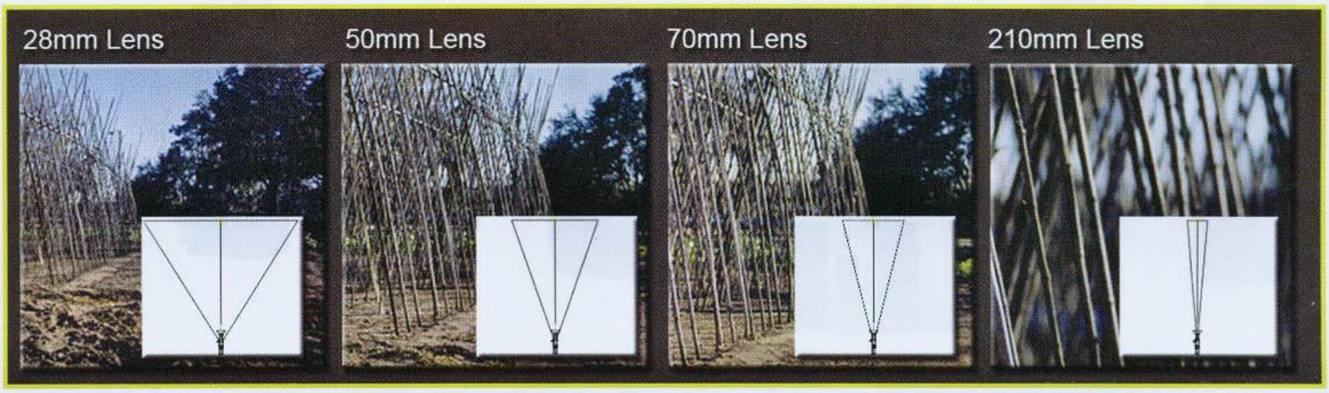
where:

h = horizontal size of the film. With a 35mm it is 36mm.
v = vertical size of the film. With a 35mm it is 24mm.



■ **Figure 6.9**
Horizontal and vertical field of view (FOV) or angle of view.

By using objectives with different focal lengths, from shorter to longer, what one obtains is a subject which gets larger with as a result less FOV.



■ **Figure 6.10**
Photograph of the same subject taken with four different objectives.

It is clear to see that in order to photograph more than one object on a film, they must all be much smaller than a single object occupying the same amount of space.

On the market there are hundreds of objectives, from small and light to heavy and expensive. We can group these into three main families: so-called "normal" objective, wide angles and teles.

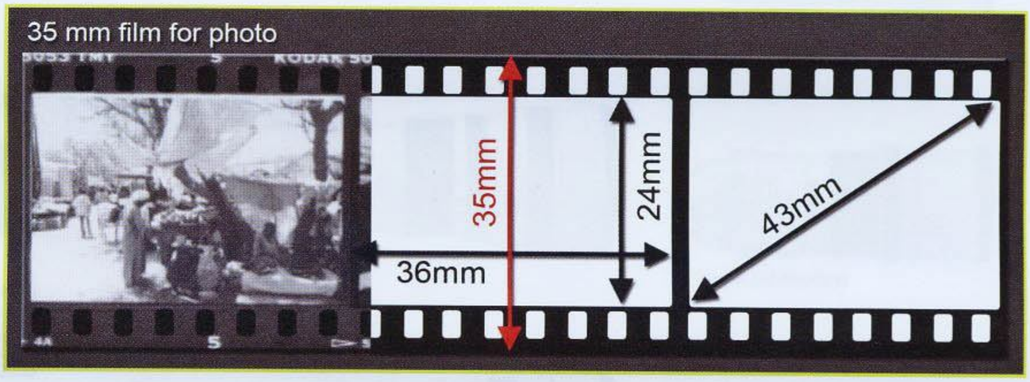
■ **Figure 6.11**
Small and medium focal length objective.

■ **Figure 6.12**
Objective with long and extremely long focal length.

Before moving on to the classification of objectives, there is a specification to be made regarding the size of 35mm film. By now it is clear that the FOV depends on two factors: focal length of the objective and the type of film used. We have already mentioned the 35mm standard quite a few times.
The film format that everybody has heard of is the 35mm, its height being in fact 35mm. In actual fact this film format was not born for photography but for cinema purposes. This is why it was endowed with small holes at the sides, which were used for the film to be attached to reel mechanisms for it to be dragged. At first frames had a size of 24 x 18mm, then with the arrival of soundtracks one had to create extra room for the audio tracks and nowadays frames measure 21 x 15mm.

■ **Figure 6.13**
A cinema film format of 35mm.

Later on, a camera was invented which could use cinematographic film, but in order to obtain images with a greater definition of details, it was projected so that frames were twice as big as the cinema version, and therefore 24 x 36mm.



■ **Figure 6.14**
A photographic film format of 35mm.

NORMAL LENS

The photographic objective is conceived to imitate the human eye and so it appears reasonable to define as normal an objective with a field of view which is more or less equivalent to the human eye. Thus one defines an objective as normal when it has a field angle of around 50° to 60°. An alternative definition which is operatively simpler defines as normal an objective which has a focal length equal to the diagonal length of the frame. Take for example the 35mm format with 24 x 36mm frame size and a diagonal length of 43.26mm. From the formula:

Field of view = 2*arctan(d/2f)

we now find the inverse formula in order to find the value of **f**, which is the focal length with a field of view of 55°.

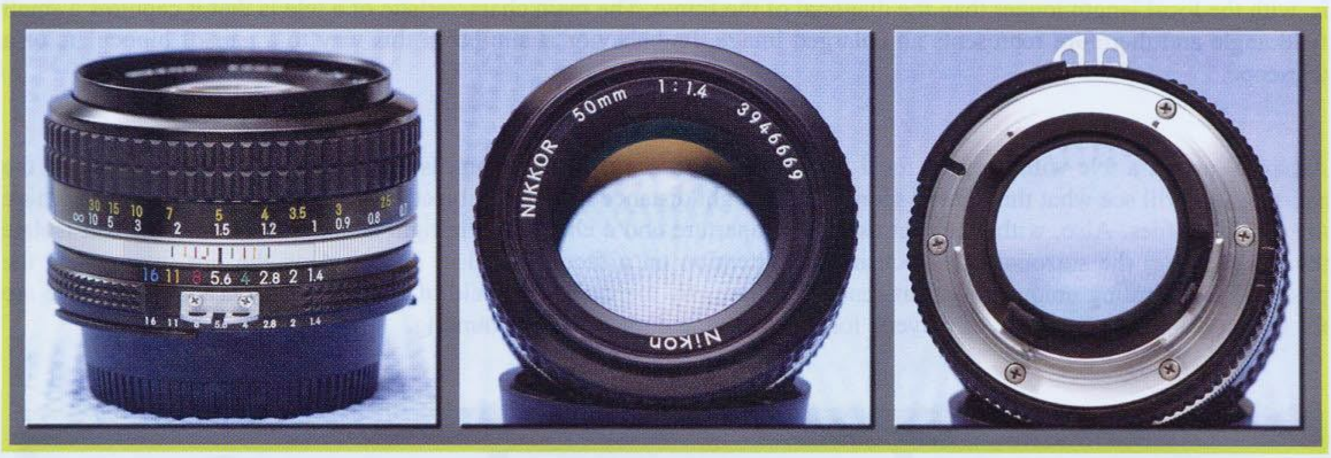
f = (d/2) * [1/(tan* field of view /2)]

where

d = 43.23mm
Field of view = 55°

From this calculation we find that in order to be defined as "normal", an objective must have a focal length of about 41.55mm. Therefore on a 35mm format, the normal objective would be 42mm. Notice that this length is hardly shorter than the length of the diagonal. Funnily enough, in the 35mm format, 50mm are considered to be normal, whereas according to the definitions we have just seen, they should actually be considered as tele objective, as they cover a field of view of 47°.

The 50mm objective is definitely not awkward and is very versatile, as it can be used for various types of photographs, from landscapes to portraits. Usually this focal length is the one provided when one purchases a camera.

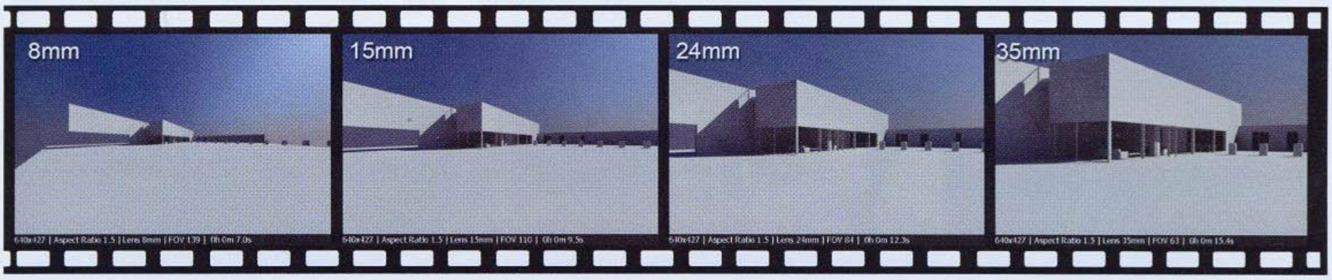


■ Figure 6.15
50mm objective.

WIDE ANGLE LENS

An objective with a greater field of view, and therefore a shorter-than-usual focal length, are called wide angle. The field of view goes from 60° - 80° for a wide angle, to reach 180° in extreme wide angle and fish eye. These last ones earn their name because of the fact that the image appears bloated because of the incredibly wide field of view, as if it were seen through a fish eye in fact. For the 24 x 36mm the most classic of them all is the 24mm, but also 35mm and 28mm are common.
The wide angle can be used for indoor or outdoor scenes. In the first of these cases, it can be useful for photographing very small and tight areas; instead outdoors it can be useful for panoramic views, architecture or everyday life situations in wide spaces. This is what happens in many journalist services. From this we can understand how the choice of one focal length as opposed to another can give a different feeling to photos.

■ Figure 6.16
Wide angle.



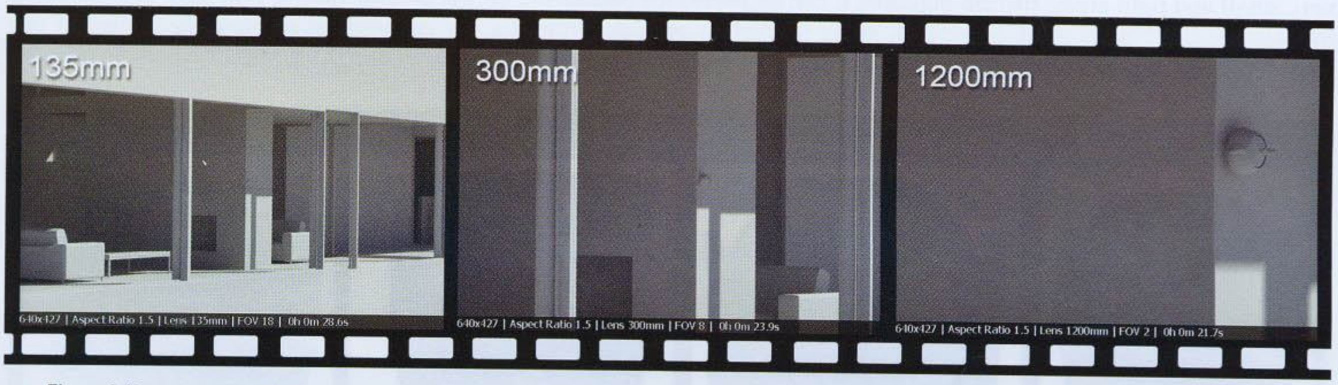
■ Figure 6.17
Render made with different wide angles.

TELE

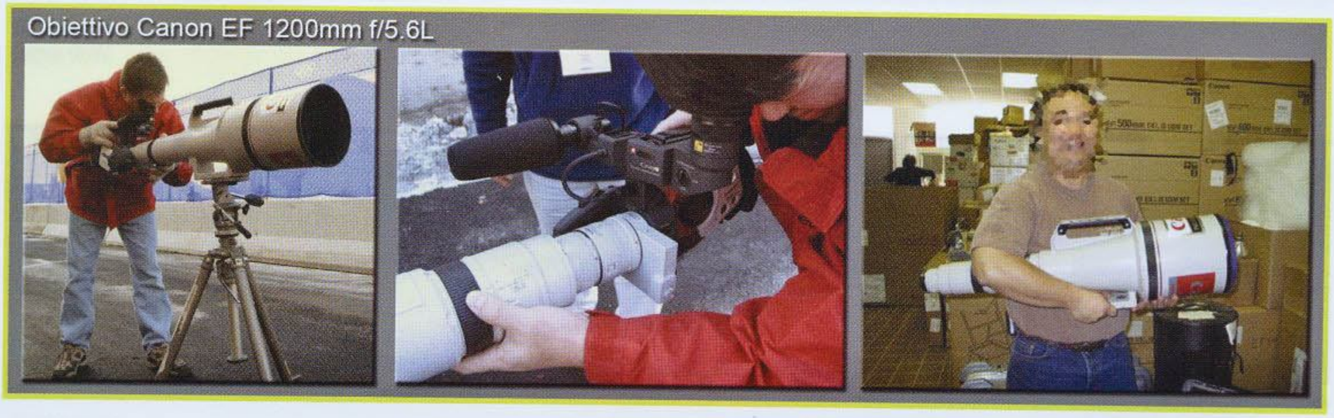
For a standard 35mm camera, a tele means it has a focal length of 70mm or more and a field of view less than 55°, that is, with the focal length longer than the diagonal of the frame. The main characteristic of a tele is that it captures a small field angle and therefore represents an enlarged image. Its behavior is similar in this way to a pair of binoculars or a telescope.

A typical use of a tele with average focal length, which is around 85-135mm, is portraits; with a correct setup of the aperture (we will see what this entails soon) and the right distance from the subject, one can obtain a correct perspective of subject profiles. Also, with a correct use of the aperture and a choice of the right shooting distance, one can isolate the subject from the surroundings, focusing the attention on a face, a hand or even just on the lips, giving one the possibility of creating unique and heart-rending pictures by managing the field of view. Other typical forms of use are for instance, for close-up shots of players, for landscapes and photographic hunting.

■ **Figure 6.18**
Some teles. Notice the difference of the 1200mm compared to the more classic and cheaper 50mm.



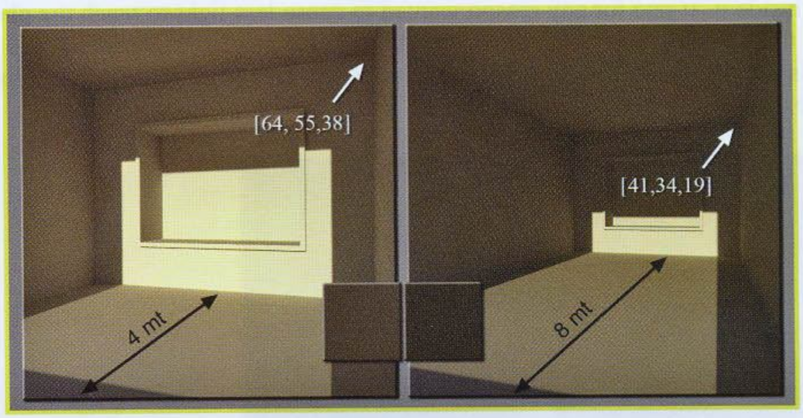
■ **Figure 6.19**
Render created with different teles.



■ **Figure 6.20**
The Canon EF 1200mm tele (or should it be called telescope?).

OBJECTIVE LUMINOSITY

Observe the following images.



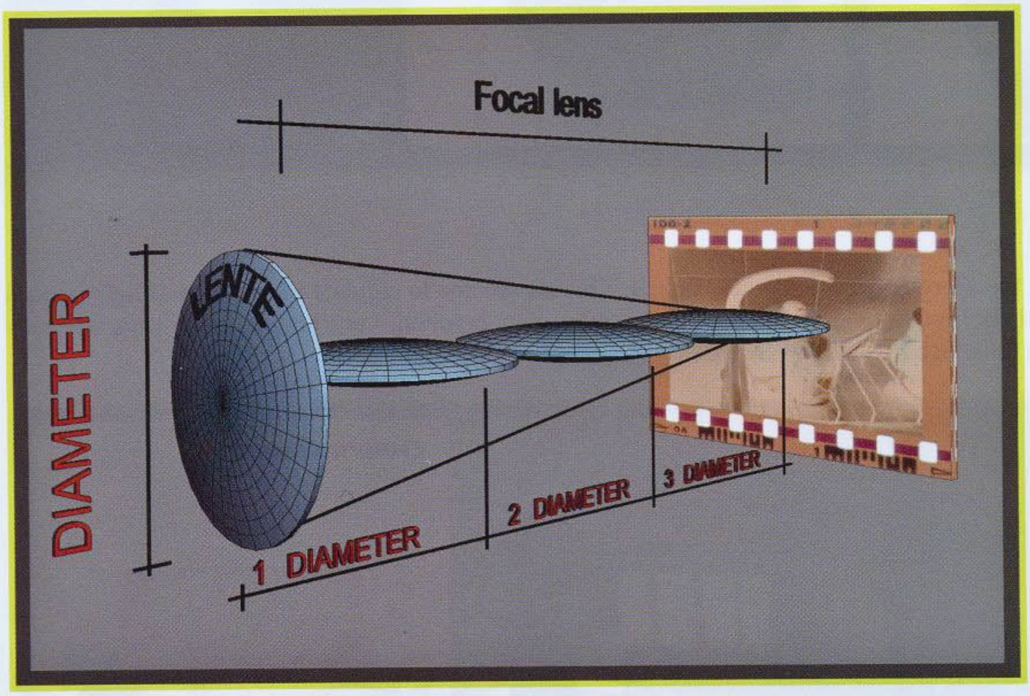
■ **Figure 6.21**
Two corridors of different lengths lit with the same amount of intensity. Notice how in the longer of the two corridors the top right corner is less illuminated and darker than the left corridor.

It is a very simple corridor, with an opening which is 4m away from the end wall, lit by sunlight. The second image represents the same room, but with twice the depth of the corridor. Here the end of the corridor is less illuminated and thus darker. This rather obvious phenomenon means that the wall closest to the corridor opening is more illuminated than the one which is deeper in. The factors which determine the amount of illumination, at equal power of the light source, are the size of the window and the depth of the corridor.

A photographic objective works in the same way and, as well as focal length, there is another property which characterizes them: luminosity, which is the maximum capacity of light transmission to the photosensitive surface. An objective has a greater luminosity; the more light it is able to transmit to the film.

If one takes the previous scene as an example, one can state that its depth can be compared to the diameter of the front lens. The following formula allows one to establish the luminosity of an objective.

LUMINOSITY = focal length: diameter.



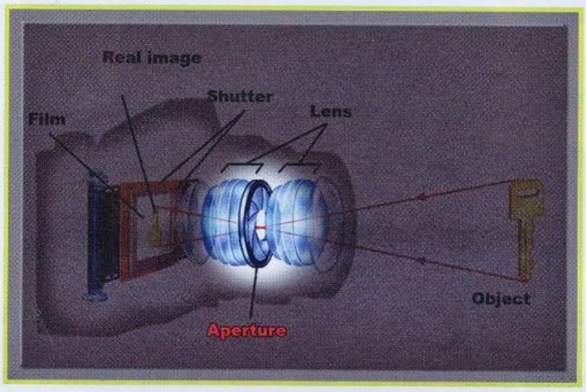
■ **Figure 6.22**
A schematic representation of the luminosity of an objective. In this case the luminosity is equivalent to f/3.

Luminosity is also called relative opening and is commonly indicated with the letter "f" (focal) followed by the bar "/" and the number which is the result of the previous division. In the example, the expression f/3 indicates that the ratio between focal length and diameter is equal to 3. Other less common ways of indicating luminosity are f.3 or 1:3.
Notice how as the denominator of the fraction increases, the degree of luminosity decreases. An f/2 objective is more luminous than an f/3 objective. At equal focal length, the bigger the lens, the more light can enter the objective, making it more luminous.

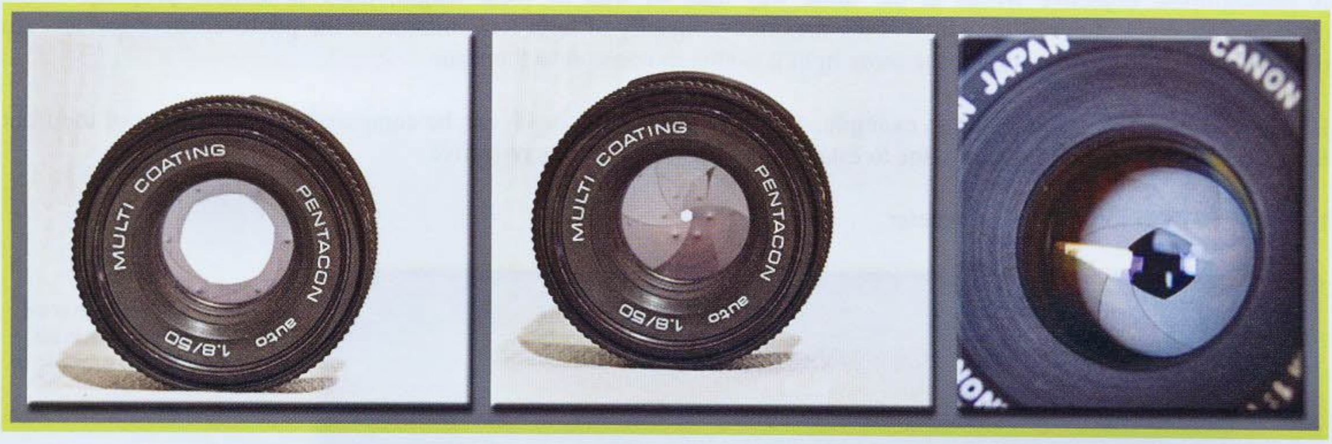
APERTURE

So far we have talked about the objective as a tube made up of a certain number of lenses which has the purpose of transmitting light to the film. One could therefore think that the amount of light transmitted is solely due to the diameter of the tube and to the lens' inside it. If one for any reason wanted to cause less light to reach the photosensitive surface, one would have to replace the objective with one which has a smaller diameter. Fortunately this is not necessary, as within the objective there is an element called aperture, which has the precise purpose of establishing the amount of light to be transmitted to the sensor.

■ **Figure 6.23**
The aperture is located within the objective.

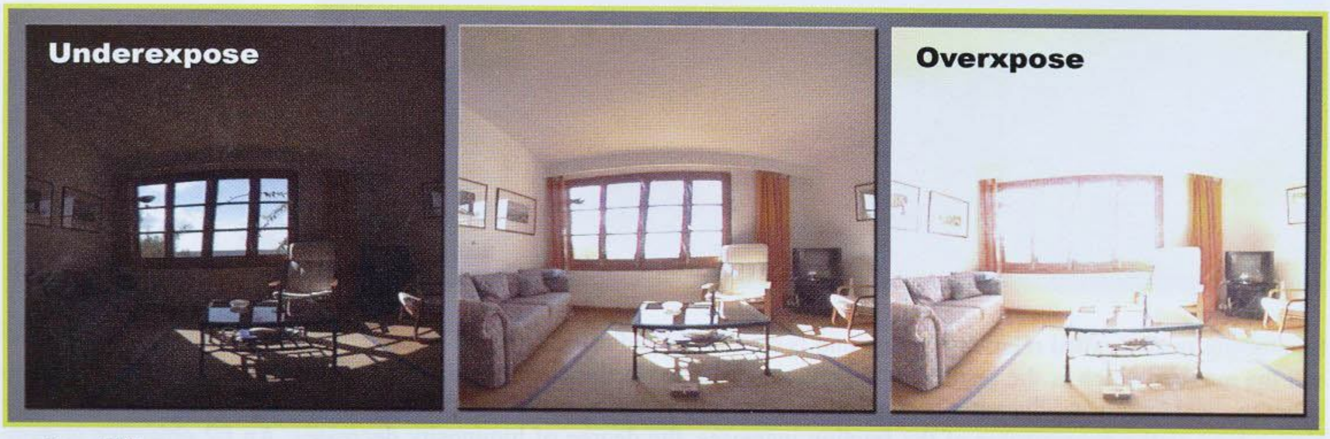


This is nothing more than a "wall" placed within the objective, made of 4, 5 or 6 blades, which can be moved to create a hole with a variable diameter.



■ **Figure 6.24**
Differenti aperture del diaframma.

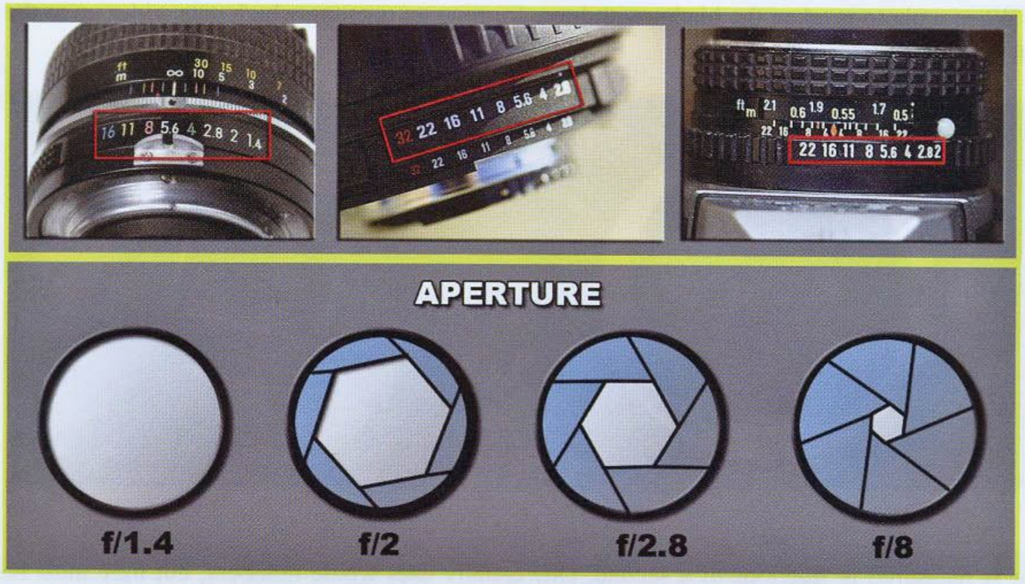
Like the human eye, so can the aperture grow wider and tighter. This allows one to regulate the amount of light needed for a the creation of a good quality photo. This happens because by photographing very luminous subjects, one may incur in the risk of letting in too much light and overexposing the image.



■ **Figure 6.25**
Photographs taken with different aperture opening amounts. If the opening is to small, the image appears dark, instead if it is too big, it appears overexposed.

If there is too little light passing through the objective, and therefore the aperture is too closed, one would obtain the exact opposite: an underexposed image.

The opening of the aperture is indicated on the metal ring of the objective by a number preceded by the letter f. These numbers are always in a standard sequence which begins with f/2 or less in some objectives and continues with f/2.8 – f/4 – f/5.6 – f/8 – f/11 f/16 –f/22 and even higher.



■ **Figure 6.26**
The ring which controls the aperture and the behavior of the blades when its value is changed.

. f/1.4, f/2 and f/2.8 - these are very open settings, good for dark environments.

. f/4 and f/5.6 - open aperture settings, for shadowy environments.

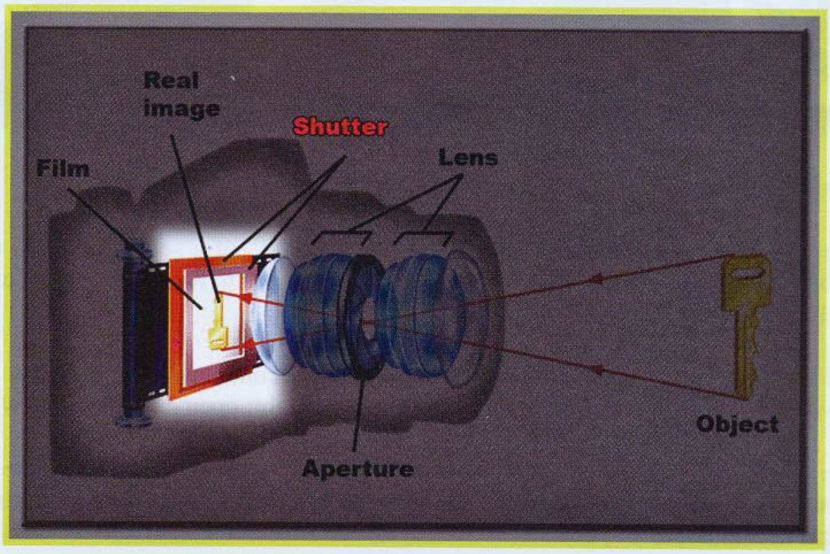
. f/8 - normal aperture used in most cases.

. f/11 and f/16 - tight aperture for bright environments.

. f/22 and f/32 - very tight aperture for extremely bright environments.

SHUTTER

The amount of light which hits the film can be checked by adjusting the aperture and changing the exposure time, in other words for how many seconds the film remains exposed by the light which penetrates through the aperture hole. This function is carried out by the shutter which works as a barrier of opacity between the light and the film, interrupted only for a strictly defined time, during the exposure of the photogram.



■ **Figure 6.27**
The shutter.

There are two types of shutters:

- **curtain shutter**: commonly used in reflex cameras, it is situated within the main camera body and is made up of two "curtains" which are extremely close to the film. They form a fissure of variable amplitude which is triggered when one takes a photograph.

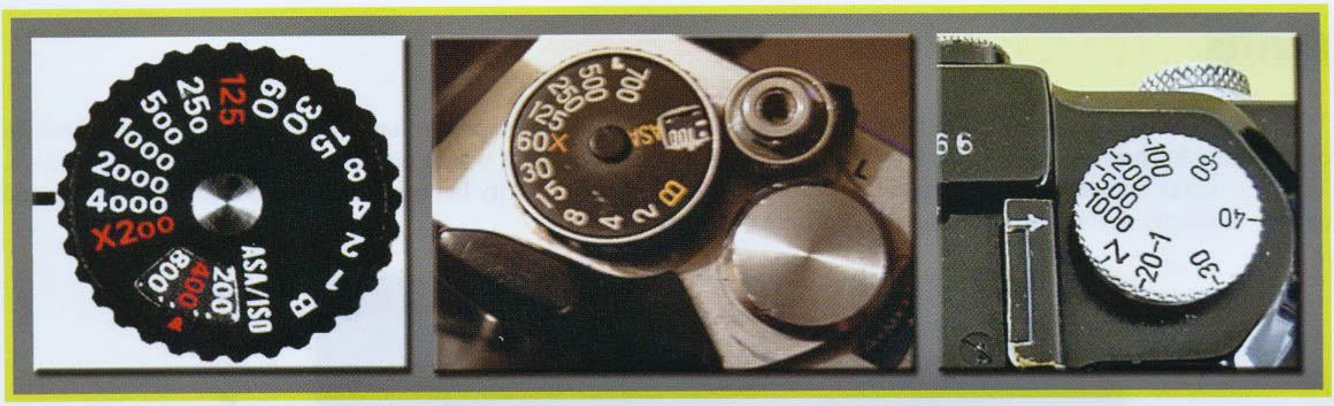
- **Central shutter**: it is located in the objective and made up of a series of blades, which form light proof disc when it is closed. When a photo is taken the blades withdraw, forming an opening which gets wider up to the maximum opening possible.

■ **Figure 6.28**
The two types of shutter: Curtain shutters and central shutters.

The value of exposure times is measured in fractions of a second. One usually uses whole numbers, but they must be considered as the denominator of a fraction. For example: 60 is read as "1/60 – one sixtieth of a second" and 250 as "1/250 – one two hundred and fiftieth of a second".
The number of these exposure times is sometimes indicated with a "t" and they are the following:

2000 – 1000 – 500 -250 – 125 – 60 – 30 – 15 – 8 – 4 – 2 – 1 – B

Obviously, similarly to f values for aperture, the bigger numbers refer to shorter exposure times and smaller numbers to longer time spans. Exposure time, that is, the time the film is exposed to light, can be set with the shutter dial or in digital devices with digital buttons or commands. The shutter dial indicates a range of exposure times which varies according to the model.



■ **Figure 6.29**
Different dials with different settings for exposure time.

. Extremely brief times are 1000 and 2000 ($1/1000^{th}$ of a second and $1/2000^{th}$ of a second), which allow even scenes with a lot of movement to be photographed without blurring. They are appropriate for conditions of extreme light: open environments with extremely direct sunlight, snow or sea.

. Brief times are from 250 to 500 ($1/250^{th}$ of a second and $1/500^{th}$ of a second), which still allow scenes with some movement to be captured without blurring. Suited for conditions of strong light: open environments with direct sunlight.

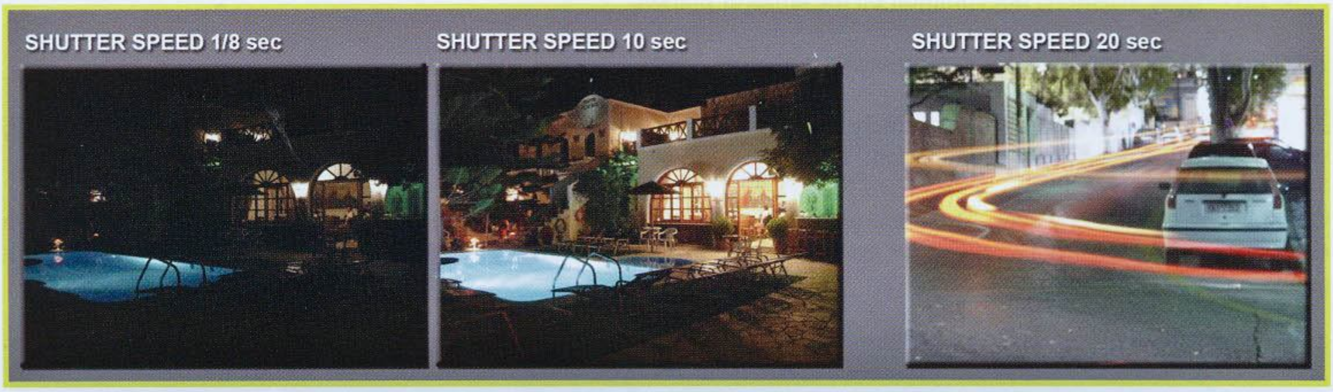
. Medium times range from 60 to 125 ($1/60^{th}$ of a second and $1/125^{th}$ of a second), which can be used without a tripod, as long as ones hand is reasonably still and the subject being photographed stay still. Apt for conditions with normal light: in the open with natural illumination.

. Long times are from 30 to 15 (1/30th of a second and 1/15th of a second), which need a tripod or support to be applied and do not allow images with movement to be photographed without them blurring. Suited for situations with weak light: indoor environments with artificial light or outdoor scenes in the shadows.

. Very long times go from 4 to 8 (1 quarter of a second and 1/8th of a second), which absolutely need a tripod or support to be applied and do not allow images with movement to be photographed without them blurring. Suited for situations with very weak light: indoor environments with very little light or outdoor scenes in very dark places.

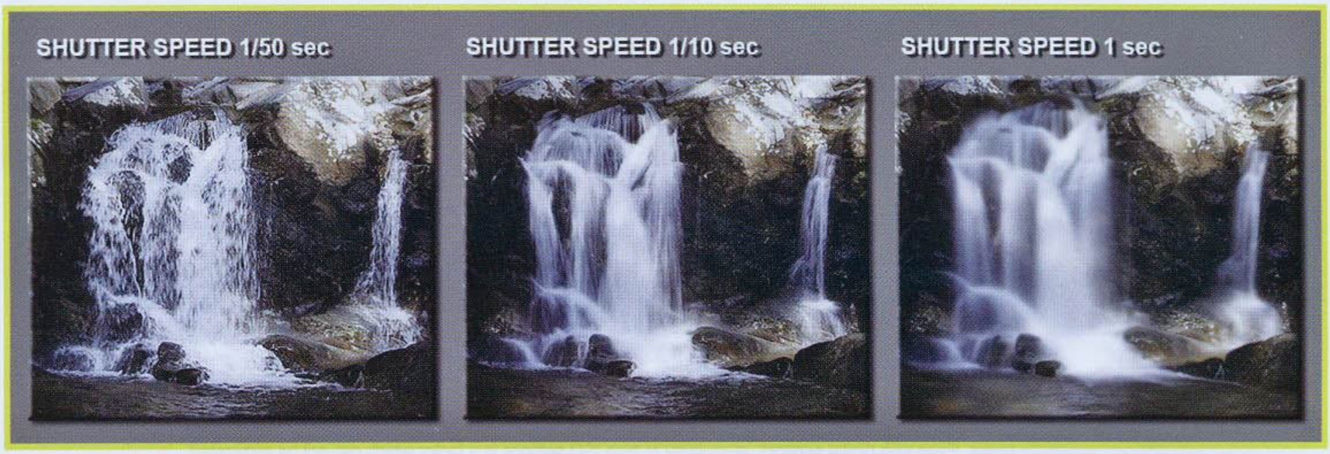
. Extremely long times are 1 and 2 (1 second and half a second), which absolutely need a tripod or support to be applied and do not allow images with movement to be photographed without them blurring. Suited for situations with very weak light: indoor environments with hardly any light or outdoor scenes very dark places or at night-time.

. The B time is the so-called bulb exposure, which is the opening of the aperture for any desired time, for as long as the photographer keeps his finger pressed on the button. This can also be for a time lasting tens of seconds.



■ **Figure 6.30**
Some examples of shutter speed.

In these last few examples one can see how the longer the shutter remains open, the brighter the image is. In the right-hand image, an exposure lasting 20 seconds has caused the trails of car lights to be captured in the photo.



■ **Figure 6.31**
Subject in movement taken with different shutter values.

In this case a waterfall has been photographed with various shutter values. With short settings the movement of water has been "frozen" so that we can see the flow of water as if stopped in time. With long exposures, the flow becomes blurry and hazy, making the photo appear much smoother.

FILM SENSITIVITY

By film sensitivity one indicates its capacity to be influenced by light. In fact some films are extremely sensitive to light and need very little for exposure, good for photographing scenes with very little light. There are also on the contrary films which are not very sensitive which require a great deal of light to be impressed, perfect for very bright scenes.

The sensitivity of films is measured with the international unit called ISO. The values are the following: 25 - 50 100 - 200 - 400 - 800 - 1600 - 3200.

. 50 ISO - the least sensitive film, good for highly illuminated photos.

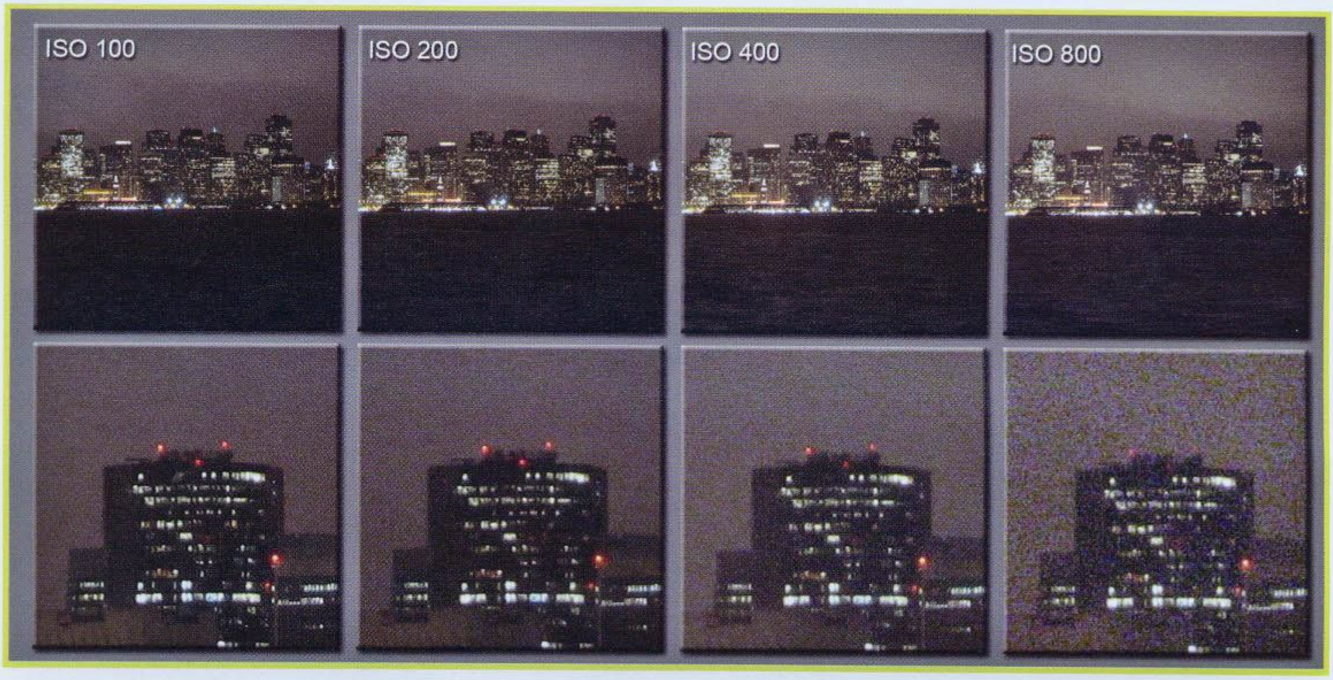
. 100 ISO - the most common type, which corresponds to the type which is commonly sold. Sometimes one may find intermediate values: 64, 125, 160 and so on.

. 200 ISO - quite sensitive, not common on the amateur market.

. 400 ISO - highly sensitive, for environments with poor light.

. 3200 ISO - this corresponds to extremely sensitive film, appropriate for photographs in places with hardly any light, without resorting to artificial illumination.

One must be warned however. The more sensitive a type of film is, the more images tend to be grainy. A photograph taken with a low sensitivity returns an image with a very low granularity, whereas a high-sensitivity photo is much grainier.



■ **Figure 6.32**
Types of film with different sensitivity levels. Even with digital sensors, granularity at high ISO settings is a factor which can not be eliminated.

When one talks about grain or granularity one refers to the fact that the image appears as if it were formed by lots of tiny dots (grains or crystals). When the grain is fine they are not visible and this makes for a good quality image where the eye recognizes areas as uniform. The grains can be seen when they become larger, and the image is therefore low quality and the eye recognizes the presence of these dots, which may vary in size. Unless one is forced to use very sensitive film, because of scarce light, it is better to use normal or low-sensitivity film, like 100 ISO or less, so that the image will be as grain-free and high quality as possible.

EXPOSURE

Every time one takes a photo, one should expose the film to an amount of light that will allow one not to have an image which is too dark or too light.

This control is obtained by taking into account the aperture, the film sensitivity, the exposure time and the amount of luminosity of the scene that one is about to photograph. Basically, every real scene, considering its luminosity, must be photographed with settings concerning:

a - film sensitivity.
b - aperture.
c - exposure time.

such that one obtains a photograph which is neither too dark nor too bright. Usually, if one uses a classic 100 ISO film and one photographs some people under the Sun, in a city square, it is probable that the right aperture value is f/11 and exposure time is 1/125. From this, one an deduce that there is no point in considering the aperture setting on its own, regardless of exposure time and both of these parameters without considering film sensitivity. Sensitivity, aperture and exposure time only make sense if they are considered as a group and if one wishes to change a value, in order to obtain the same amount of light entering the objective, one must necessarily adjust the other two parameters accordingly, so as to maintain the right balance.
At this point, we shall introduce a new element, the EV. The EV exposure value is the value of light intensity to which combinations of aperture and shutter time correspond, allowing the same amount of light to reach the film. One conventionally defines the EV as a logarithm in base 2:

$EV = \log2(A^2/T)$

where **A** is the opening amount of the aperture and **T** is exposure time. At different combinations of time and aperture there are equal EV values. For instance f5.6 and 1/60 is equivalent to f8 and 1/30. For instance EV 0 corresponds to the pair aperture f/1.0 and exposure time = 1 second. As we know that each following aperture setting halves luminous intensity, in order to obtain the same EV 0 value with an aperture of f/1.4, one must double exposure time and bring it up to 2.
EV value is always referred to a sensitivity of 100 ISO. If one must by any chance change the film speed, one must compensate for this with a different time/aperture pair. If for example one changes from 100 ISO to 200 ISO one must halve the exposure time or aperture amount.

	f-number												
Time (s)	**1.0**	**1.4**	**2.0**	**2.8**	**4.0**	**5.6**	**8.0**	**11**	**16**	**22**	**32**	**45**	**64**
60	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
30	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
15	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
8	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
4	-2	-1	0	1	2	3	4	5	6	7	8	9	10
2	-1	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	3	4	5	6	7	8	9	10	11	12
1/2	1	2	3	4	5	6	7	8	9	10	11	12	13
1/4	2	3	4	5	6	7	8	9	10	11	12	13	14
1/8	3	4	5	6	7	8	9	10	11	12	13	14	15
1/15	4	5	6	7	8	9	10	11	12	13	14	15	16
1/30	5	6	7	8	9	10	11	12	13	14	15	16	17
1/60	6	7	8	9	10	11	12	13	14	15	16	17	18
1/125	7	8	9	10	11	12	13	14	15	16	17	18	19
1/250	8	9	10	11	12	13	14	15	16	17	18	19	20
1/500	9	10	11	12	13	14	15	16	17	18	19	20	21
1/1000	10	11	12	13	14	15	16	17	18	19	20	21	22
1/2000	11	12	13	14	15	16	17	18	19	20	21	22	23
1/4000	12	13	14	15	16	17	18	19	20	21	22	23	24
1/8000	13	14	15	16	17	18	19	20	21	22	23	24	25

■ **Figure 6.33**
EV chart for a sensitivity of 100 ISO.

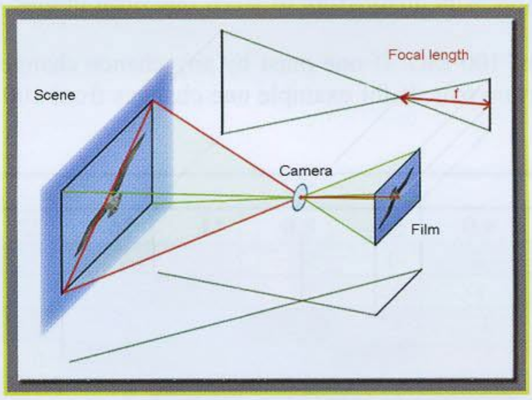
Luckily most modern cameras contain a measuring system called a light meter, which informs one of whether the adjustment of aperture/exposure time is correct for the amount of light and film sensitivity.
The light meter is a machine able to detect the amount of light and translate it in light value. From this one can work out the aperture/exposure time settings to apply. Basically, what was once done manually (setting these two values) according to the photographer's experience, is now done with the help of this device. If the camera is automatic, the light meter measures the light and sets the two values, instead if the camera is manual it will just notify the user of the light received with the set exposure time and aperture.

■ Figure 6.34
Two different models of digital light meters.

At this point the obvious question is: what is the right criterion by which to choose objective, aperture, exposure time and film settings? Before answering this question, one must add the last missing jigsaw piece before taking our photo. We are talking about the Depth of Field (DOF).

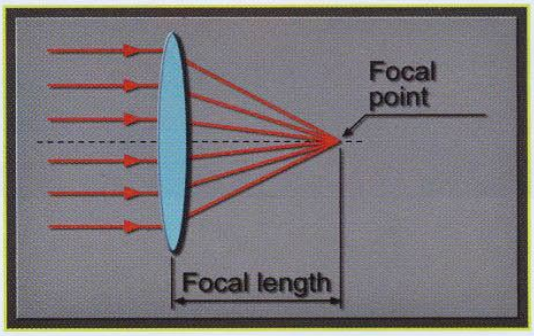
DEPTH OF FIELD (DOF)

We have said that the focal point in a lens is the point along the optical axis where the objective is able to reconstruct a perfectly focused image. The film is placed exactly in this point. In actual fact more than focal point, we should call it the focal plane, as the lens reproduces an overturned rectangular image.



■ Figure 6.35
The focal length in an optical device is the distance between the focal plane and the lens.

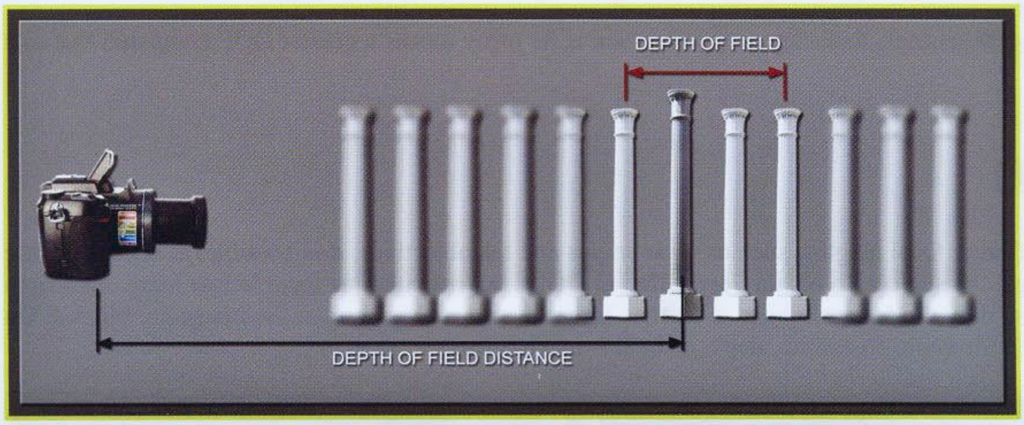
Now imagine a hypothetical situation whereby the lens is hit by perfectly parallel rays of light. After passing through the lens, the rays of light are deviated and converge in one spot. Its position depends on the curvature of the lens. This spot is called the focal distance of the lens, and it is a value used to classify objectives in order to compare them to each other.
The condition of parallel rays is referred to the fact of trying to focus something infinitely far away, and it is used to provide a focal value which is common to all lenses and allows one to compare them. In actual fact, the rays of light from the objects to be photographed hardly ever reach the camera parallel to one another and this requires the objective to be focused.



■ Figure 6.36
The rays of light coming from infinitely far away reach the lens parallel to each other.

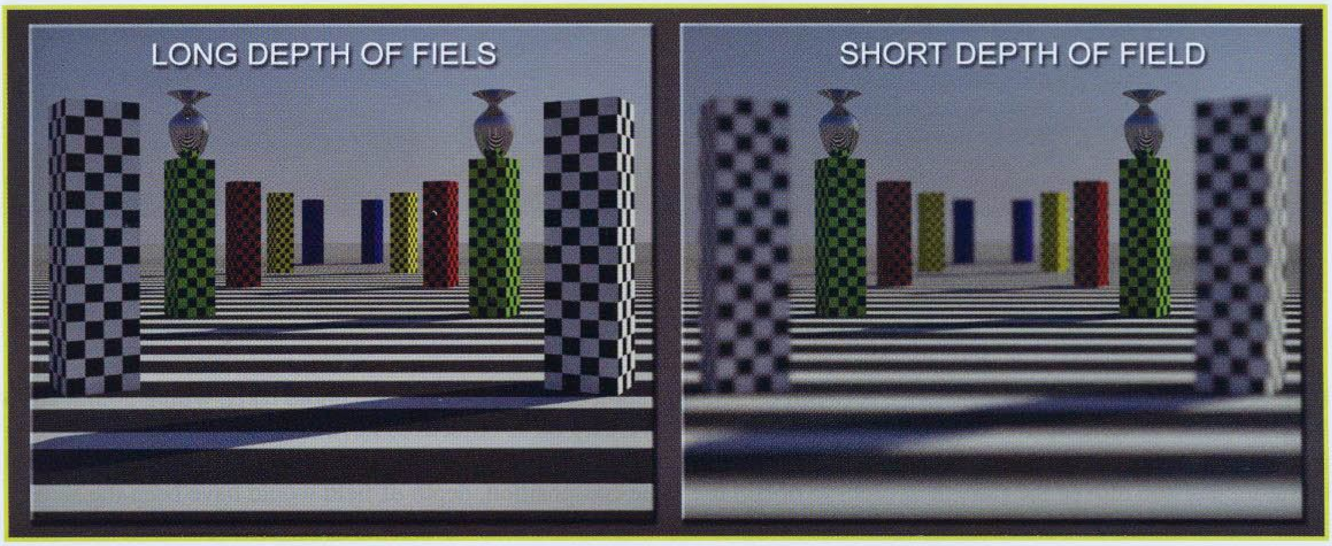
The rays of light hitting the objective are deviated in such a way that they concentrate in the focal point of the lens, a short distance past the lens itself. If the film is located in the focal point, the image which is seen after development is perfectly focused. Instead if the focal point falls before or after the film surface, the photo becomes blurred. One might then suggest putting the film exactly in correspondence with the focal point and the image will always be focused. This would be correct, were it not for the fact that the focal point changes!

At this stage, the only thing we know for sure is the position of the focal point. It is located at a distance from the lens which is equivalent to the focal value of the lens itself, but only if rays hit the objective completely parallel. With this condition, we know that if we want to focus an objective of 4mm, it is enough to place the film 4mm away from the lens. The only problem is that the only objects capable of reflecting perfectly parallel rays are extremely far away. The Sun for example, does. The mountains on the horizon as well. In photographic jargon, these far away objects are nicknamed "infinite". Everything closer than infinite reflects rays with variable incident angles. For these objects, the focal point no longer corresponds with the characteristic focal value of the objective but is located elsewhere. One could also say that rays reaching the lens in a parallel manner from far away objects converge in a point which is equivalent to the focal value of the objective, whereas rays coming from closer objects are concentrated beyond the lens, at a variable distance according to how near they are to the objective. Adjusting the focus of the objective consists of moving the objective backward or forward so as to bring the CCD in correspondence of the focal point of the subject or the area of interest. One will no longer be focusing infinity but a closer object, which will become the center of the focus. The depth of field is the distance before and beyond the main subject which still appears focused.



■ **Figure 6.37**
Depth of field.

An extended depth of field means a wide focused area: this means that, as well as the subject being focused, there are many planes after and before the focal point. This way one can maintain more than one subject focused, even if they are at different distances. A small depth of field means that the area where objects appear focused only very near the plane of the subject, whereas almost all the other planes are blurred. This way we have a selective focus which makes the main subject stand out.



■ **Figure 6.38**
A low depth of field means only focusing a small part of the image, leaving the remainder out of focus.

There are three elements which determine the quantity of the depth of field:

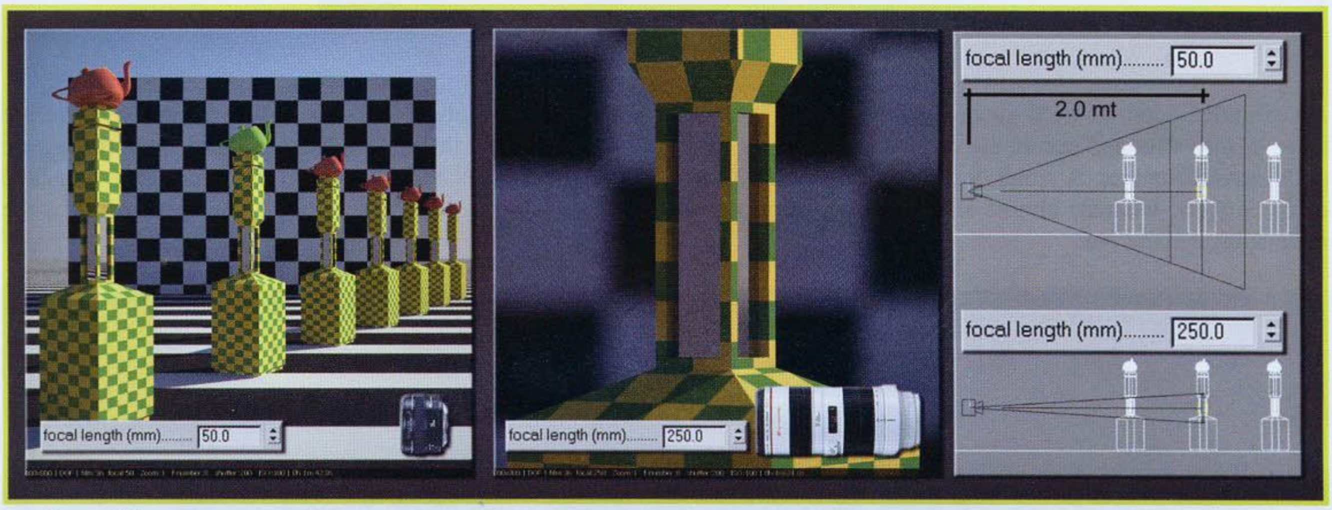
. Focal length
It is commonplace to say that objectives with a greater focal length, such as teleobjectives, have a smaller DOF, enabling them only to focus a small portion of the subject. This statement requires a slight correction, because the ratio between DOF and focal length is a consequence more of the typical use that one makes of different focal lengths, than of the actual physical properties of lenses. Long focal lengths for far away objects, short focal lengths for closer subjects. This concept can be clarified with an example. Consider a photographer using a 400mm for taking a photograph of a bird which is 10m away. With an aperture of f/2.8, the DOF is 10cm. If the same photographer changed his objective to 50mm, the DOF would change to 7.62m, which confirms the statement we have made about the ratio between DOF and focal length. However, if the photographer wanted to recompose the image so that the bird occupies the same space in the frame as before, he would have to move closer to the subject up to a distance of 1.25m. At this point the DOF would go back to being exactly the same as before, that is, 10cm.

. Aperture
With a closed aperture (f/32), the depth of field is extended, in other words the background and subject are focused. With an open aperture (f/5), DOF is reduced. As a result, the subject is focused, but the background remains blurred. High aperture settings correspond to low DOFs and vice versa. In most cases, objectives give the best results with medium aperture settings.

. Distance from the subject
At constant rates of all the other settings, focusing an object which is far away means a greater DOF compared that of a closer subject.

DOF AND FOCAL LENGTH

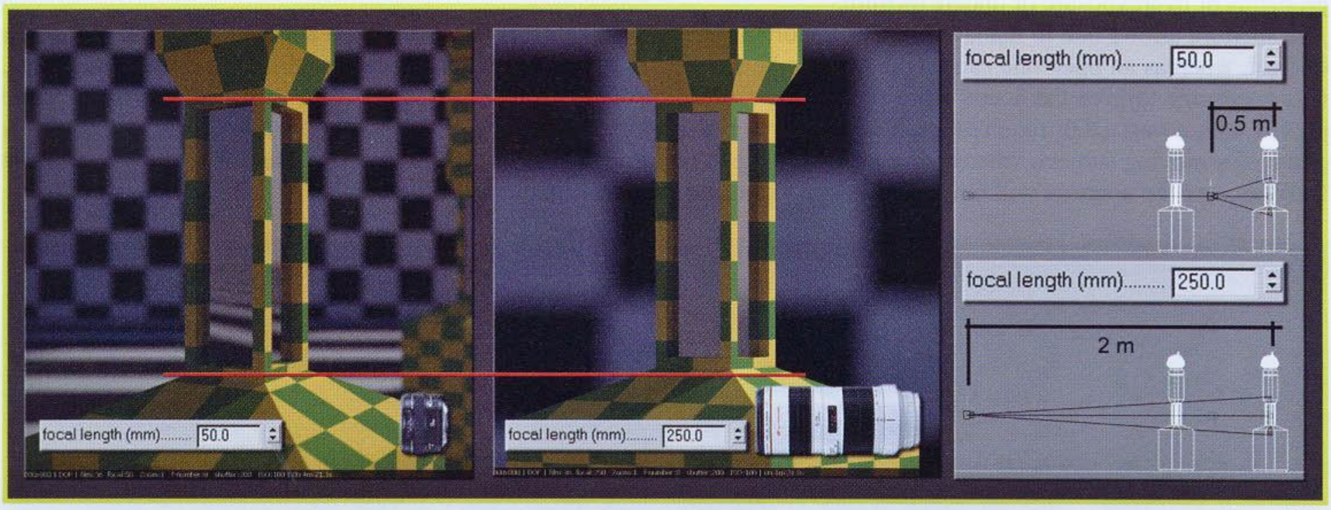
By using objectives with different focal lengths, such as wide angles and teleobjectives, it is possible, at equal distance between the photographer and the subject, to obtain very different DOFs. In particular, as focal length increases, depth of field decreases. In this case only the focused object is clear, whereas the remaining parts of the image are blurred. On the contrary, with normal or wide angle objectives, depth of field is much greater.



■ **Figure 6.39**
DOF difference according to focal length variation.

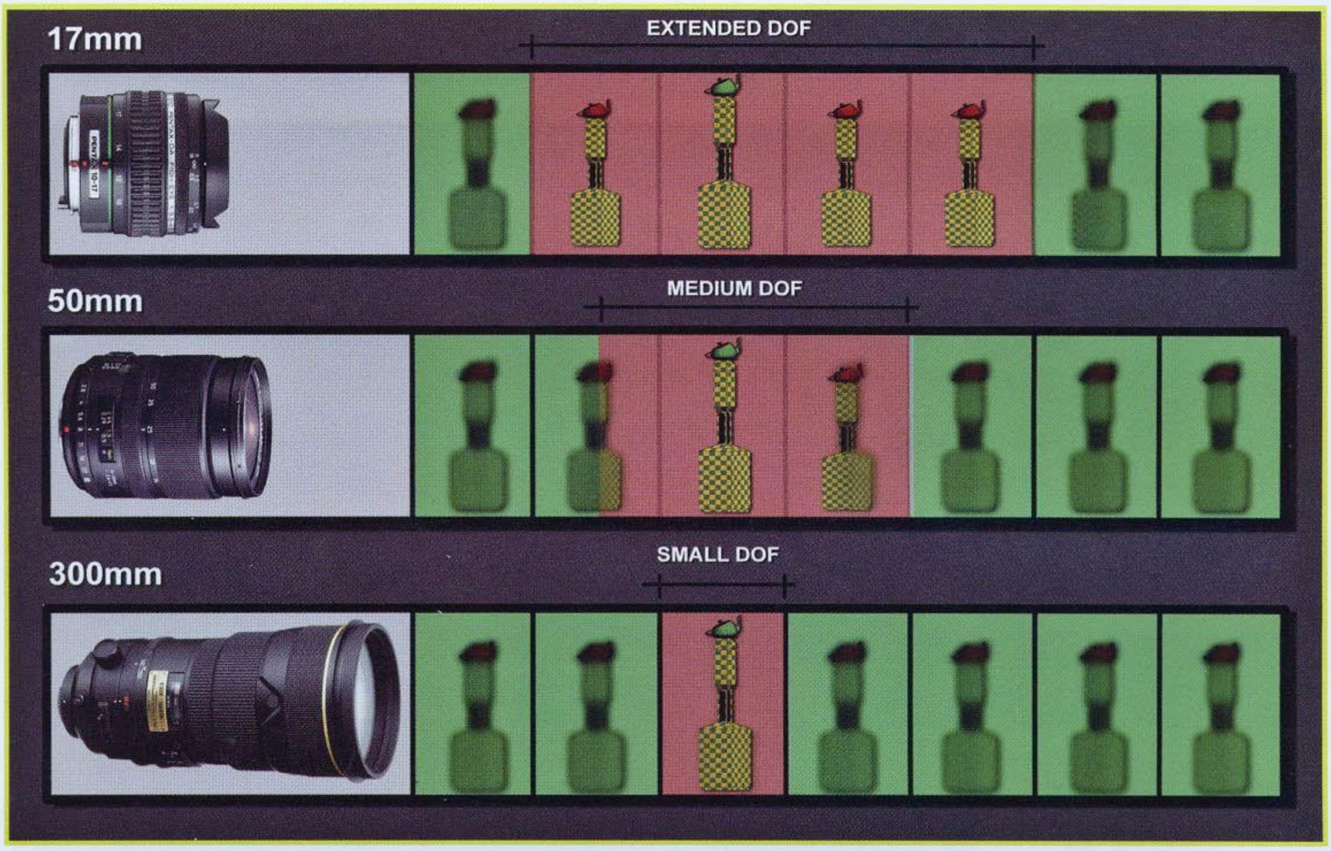
In both the previous photos, the distance between camera and subject is of 2 meters. The photo taken with a 50mm has a lot more depth of field than the one taken with a 250mm. Obviously these objectives have extremely different shooting angles, so also the composition and size of the object changes a lot. Instead if the size of the object were the same and therefore one had moved the camera in order to obtain the same composition, depth of field would have been the same.

In the next image the photographer-subject distance obtained with a 50mm objective has been changed. To be precise, the camera has been placed closer to the geometry, so as to obtain an object of the same size as was obtained with the 250mm objective. The 250mm seems to have a more blurry background, but the difference is due to the different visual angle of the two objectives and not because of the DOF, which is exactly the same.



■ **Figure 6.40**
DOF variation caused by the focal length being changed. Notice the fact that there is less perspective distortion because of the use of a teleobjective.

One can therefore easily ascertain that the depth of field diminishes with the increase in focal length; so a teleobjective with a long focal length provides images with a small DOF, instead a short-focal objective, such as a wide angle, produces images with a significantly extended DOF.

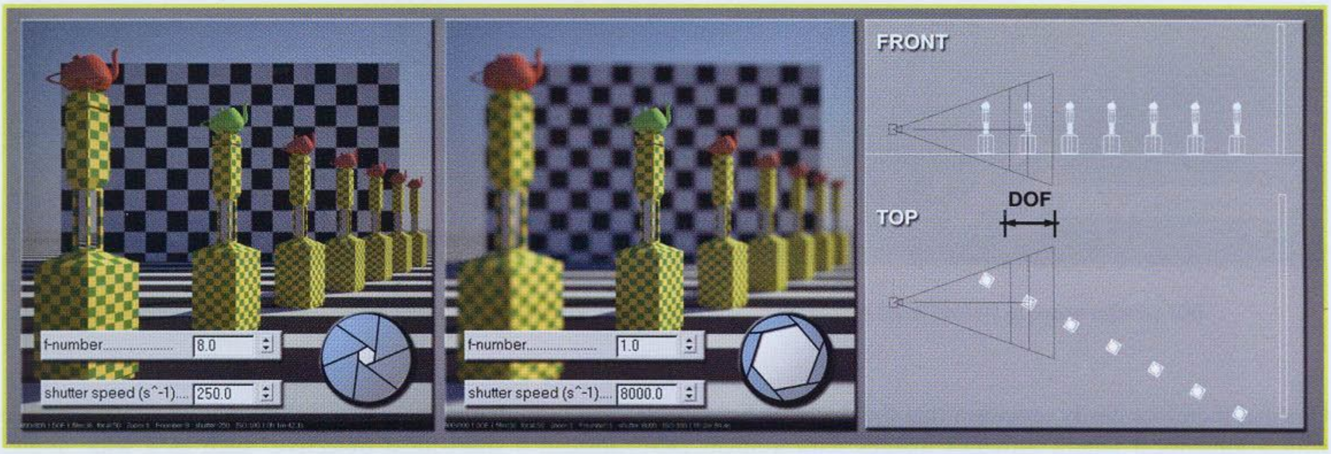


■ **Figure 6.41**
Variation of the DOF according to the type of objective used.

DOF AND APERTURE

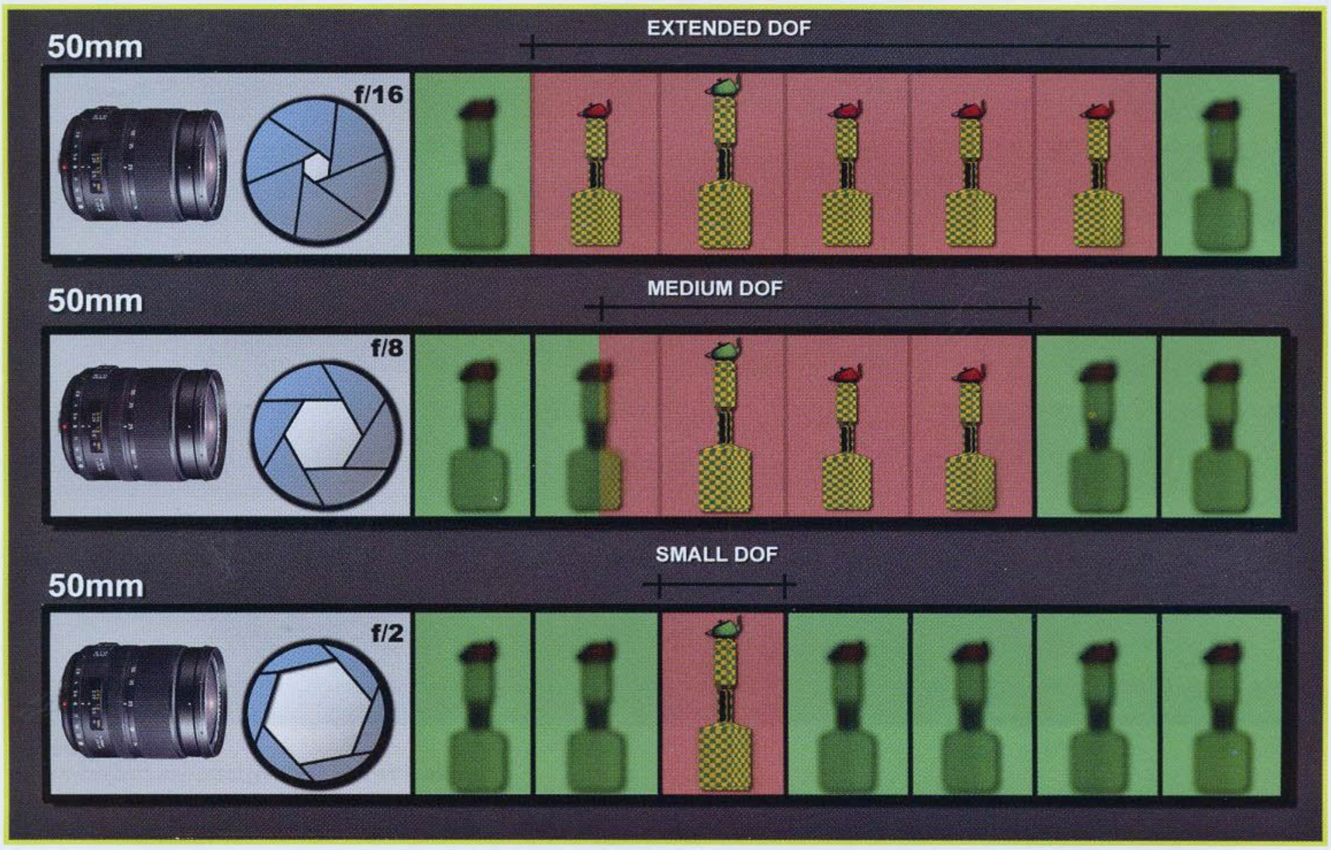
The aperture is an opening, usually circular or polygonal, through which light flows when going through the objective in order to reach the photosensitive surface. By regulating it, one can obtain properly exposed photographs, without problems of overexposure or underexposure.

The aperture allows one to manage another important feature: the depth of field. A closed aperture creates a vast depth of field, whereas an open one causes the DOF to be smaller. In order to compensate for the different opening amounts, it is necessary to give the scene more or less exposure. For this reason, a closed aperture which allows one to take photos with a vast DOF requires a longer shutter speed in order to achieve an adequate exposure. This means that a photo of a moving subject, where a short shutter time is required so that the image does not blur, does not allow a vast DOF because the aperture will have to be opened a lot and exposure time must be very short.



■ **Figure 6.42**
· Variation of the DOF due to the use of different apertures.

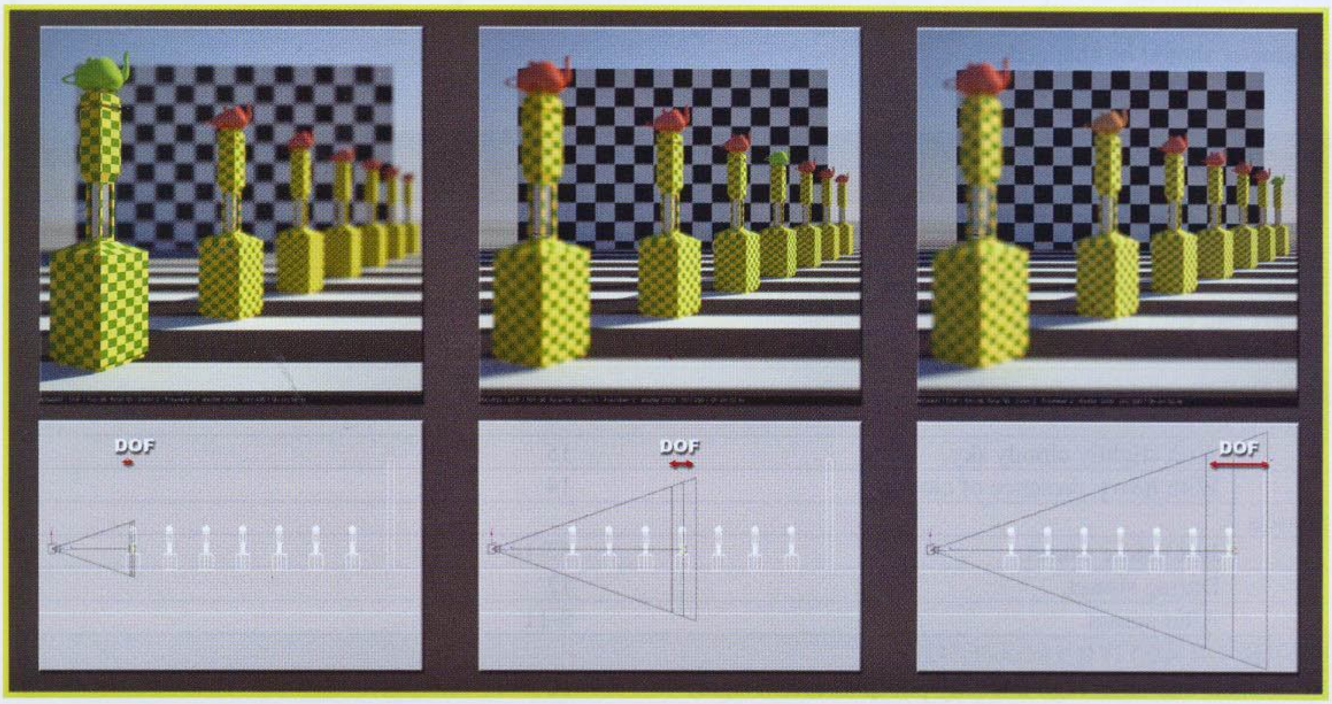
In the next image one can observe the variation of the depth of field on the basis of the degree of aperture opening.



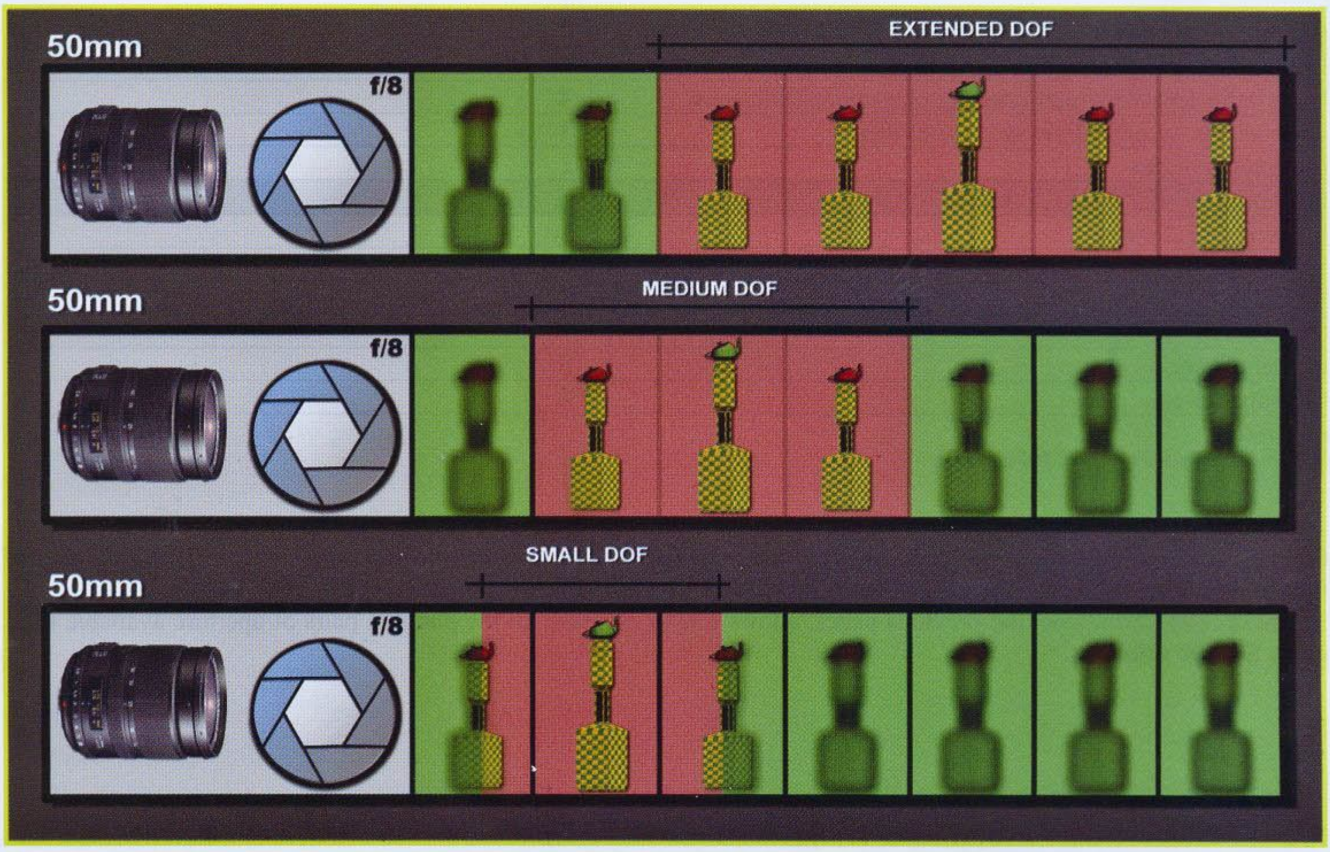
■ **Figure 6.43**
Behavior of the DOF according to different aperture values.

DOF AND SHOOTING DISTANCE

The last parameter which can influence the depth of field, as well as the focal length and aperture, is the distance of the photographer from the subject. In this case both the aperture and the focal length are fixed. The only element which varies is the distance between the camera and the subject. By observing the following images we can ascertain that when one photographs a very close subject, the depth of field is always smaller than it is when the same subject is considered at a greater distance.



■ **Figure 6.44**
By means of the screenshots one can see the behavior of the ***VRayPhysicalCam***, which allows one to define the DOF directly in the main window and emphasizes the increase in DOF as a function of a greater distance between the photographer and the subject.



■ **Figure 6.45**
The DOF and the shooting distance.

CORRECT EXPOSURE

At this point we know al the variables which come into play when a photo is taken. One might ask which settings are best in order to take a photograph of a snowy landscape, a room or a sunset. There are no mathematical rules for this and a lot depends on what the photographer wishes to fulfill, but it is however possible to create a chart which allows one to determine, for a particular subject, the correct EV value when using a classic ISO 100 film. Another very common film is the ISO 400, suited for scenes with very little illumination. With different values, obviously, the EV parameter should be changed.

LIGHT CONDITION	EV_{100}
DAYLIGHT	
Sand of bright snow directly illuminated by the Sun.	16
A daytime scene with no clouds.	15
Slightly cloudy daytime scene with a few mild shadows.	14
Cloudy day with no shadows.	13
Very cloudy day.	12
OUTDOOR NATURAL LIGHT	
Rain	
With slightly cloudy sky.	15
With heavy presence of clouds.	14
Sunsets	
Just before sunset.	12-14
During sunset.	12
After sunset.	9-11
OUTDOOR ARTIFICIAL LIGHT	
Neon lights and other high intensity phosphorescent signs.	10
Illumination for night-time sports activities.	9
Burning buildings.	9
Illuminated streets and roads.	8
Christmas lights.	7
Buildings, monuments and fountains.	7
Panoramic views of illuminated buildings.	5
Neon lights and other high intensity phosphorescent signs.	4
Illumination for night-time sports activities.	3
Burning buildings.	2
INDOOR ARTIFICIAL LIGHTS	
Illuminated galleries.	10
Sports events, exhibitions.	9
Sports halls.	8
Offices and work areas.	7
Household areas.	5
Christmas trees.	4

■ **Figure 6.46**
Exposure values for a film with ISO 100 sensitivity.

Once the exposure of a scene has been defined, one must fix the two aperture and exposure settings. The sensitivity of the film is a fixed value.
The next chart, taken from the previous pages, allows one to establish all the possible pairs of aperture available for a given EV 100 value.

	f-number												
Time (s)	**1.0**	**1.4**	**2.0**	**2.8**	**4.0**	**5.6**	**8.0**	**11**	**16**	**22**	**32**	**45**	**64**
60	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
30	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
15	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
8	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
4	-2	-1	0	1	2	3	4	5	6	7	8	9	10
2	-1	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	3	4	5	6	7	8	9	10	11	12
1/2	1	2	3	4	5	6	7	8	9	10	11	12	13
1/4	2	3	4	5	6	7	8	9	10	11	12	13	14
1/8	3	4	5	6	7	8	9	10	11	12	13	14	15
1/15	4	5	6	7	8	9	10	11	12	13	14	15	16
1/30	5	6	7	8	9	10	11	12	13	14	15	16	17
1/60	6	7	8	9	10	11	12	13	14	15	16	17	18
1/125	7	8	9	10	11	12	13	14	15	16	17	18	19
1/250	8	9	10	11	12	13	14	15	16	17	18	19	20
1/500	9	10	11	12	13	14	15	16	17	18	19	20	21
1/1000	10	11	12	13	14	15	16	17	18	19	20	21	22
1/2000	11	12	13	14	15	16	17	18	19	20	21	22	23
1/4000	12	13	14	15	16	17	18	19	20	21	22	23	24
1/8000	13	14	15	16	17	18	19	20	21	22	23	24	25

■ Figure 6.47
Chart for the definition of the exposure/aperture pair on the basis of any given EV 100 value.

If one uses a film with a greater sensitivity, one can use apertures which tend to be more closed or brief exposure times, whereas a less sensitive film leads us to use more open apertures and longer exposure times. To preview how many exposure or aperture settings one loses or gains by changing sensitivity, one must see how much it changes from ISO 100. This calculation is very easy with ISO values because by doubling them one halves times and vice versa. To give a practical example, if one has a luminous intensity of 14 EV and a film of ISO 100, one can use the pair "f/32" and "t1/15". If we change to a film of ISO 400, four times as sensitive, one can use an exposure time which is 4 times as fast and therefore we have the values "f/32" and "t1/250" or one can use an aperture which is 4 times as small, giving us "f/8" and "t1/15".

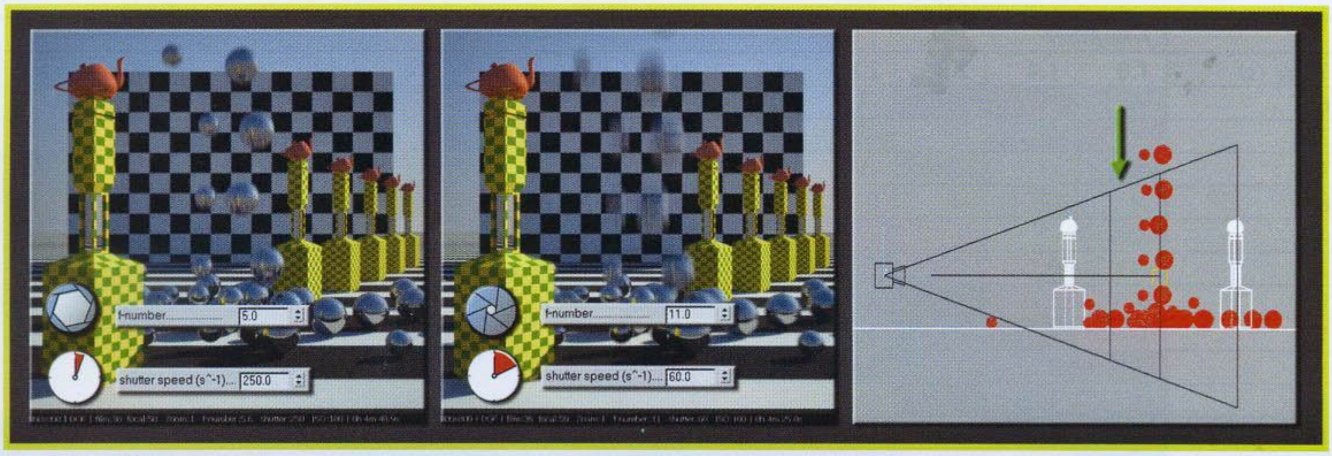
Once the EV has been defined, three factors influence the choice of the aperture setting.

MOVEMENT

If the subject or the person taking the photo is moving, one must consider that if one wishes to "freeze" the subject, one must use a fast exposure time, in relation to the focal length of the objective one is using and to the speed of the subject itself. Otherwise the object will appear blurry which will give it a sense of movement, the so-called motion blur effect. The following chart indicates the suitable times for freezing a moving object; the chart is an approximation of exposure times used in recurring photography situations. On the contrary, in order to fulfill motion blur effects and give the feeling of movement, one should widen the setting by two units.

SUBJECT	FRONTAL MOVEMENT	DIAGONAL MOVEMENT	TRANSVERSAL MOVEMENT
Person walking	1/30	1/60	1/125
Slow traffic	1/60	1/125	1/250
Boats with veils	1/60	1/125	1/250
Cyclists	1/60	1/125	1/250
Motorboats	1/125	1/125	1/500
Trains	1/125	1/250	1/500
Athletics	1/250	1/500	1/1000
Football matches	1/250	1/500	1/1000
Car racing	1/250	1/500	1/1000
Low speed airplanes	1/250	1/500	1/1000

■ Figure 6.48
Opening values for different moving subjects.



■ **Figure 6.49**
The effect of exposure time and aperture on ***motion blur***.

In the previous image a series of chrome spheres fall from above. With adequate exposure time and aperture settings one can manage the "trail effect" efficiently.

DEPTH OF FIELD

If one wishes to isolate the subject from the surroundings, one must use long focal lengths and open the aperture, or with medium focal lengths, move closer to the subject. Vice versa, if one wishes to have most of the image focused, one must close the aperture by a certain amount or use a wide angle. Clearly the choice of the objective does not influence only the focus, but also affects the image perspective.

DISTANCE FROM THE SUBJECT AND CHOICE OF FOCAL LENGTH

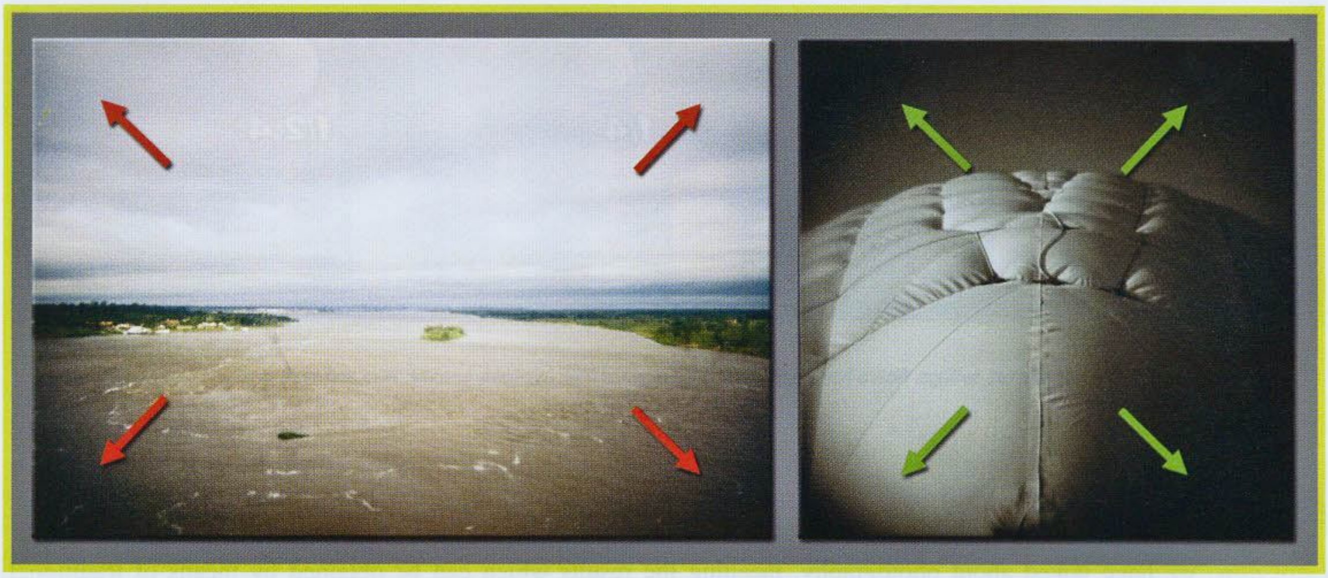
In this case, there is no precise rule for choosing the focal length, but in general one can use the following chart in order to decide which type of objective to use on the basis of the subject to be photographed:

SUBJECT	OBJECTIVE
Landscapes and panoramic views	19mm
Landscapes and indoor photographs	24mm
Landscapes and indoor photographs	28mm
Monuments, architecture and documentaries.	35mm
Architecture and general subjects.	50mm
Portraits and documents.	85mm
Portraits and animals.	105mm
Portraits, animals and sports photos.	200mm
Portraits, animals and sports photos.	300mm
Portraits, sports photos and ornithological photos.	400mm

The information shown is the basic knowledge for correct photographs. There are however other particularities connected with a camera. More specifically, we will now face the subject of the vignetting effect and the bokeh effect, a word we have already encountered during the description of the ***VRay:Camera*** rollout parameters.

VIGNETTING EFFECT

Vignetting is the phenomenon which indicates the reduction of luminosity of the image compared to the center. It is a defect found in photography due to poor quality optical elements or the use of blinds which are not apt for the focal length of the objective applied.



■ **Figure 6.51**
Vignetting is visible along the four corners of the images.

There are three types of vignetting:

.mechanical vignetting: it is an effect which depends on the aperture and can be compensated for by increasing the opening of the objective aperture.

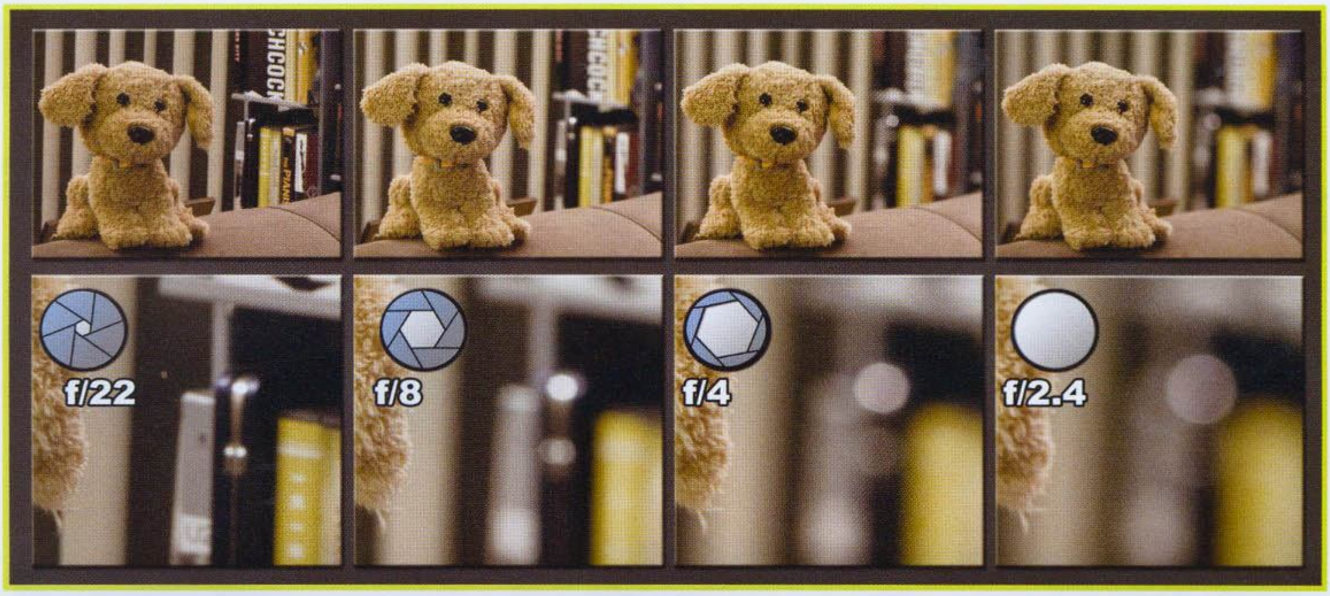
.optical vignetting: the attenuation of light which is inherent to the lens. Wide angles and compact camera lenses tend to have this problem. This cannot be made up for with different aperture settings. In order to correct it, one must use a gradual filter or elaborate the image during post-production with a photo retouching program.

.digital vignetting: this is only present in digital cameras. Most digital cameras automatically compensate both digital and optical vignetting when the convert raw data from the sensor into the standard format.

BOKEH EFFECT

This term refers to the quality of an image in the area which is in the out-of-focus or blurred area. With any probability one might think that a blurred image is altogether no different from any other one, but this is not so. The appearance of a blurred image can be quite precise, and can improve or hinder image quality. Bokeh is extremely important in Japan, to the point that some photography magazines include the results of the tests on objectives regarding this aspect alongside other criteria concerning performance evaluation. The problem, at least for the moment, is that the Bokeh effect is purely subjective. No one so far has established a way of measuring this effect with a reliable method, making it an understandable sign of good quality.
German optical products are considered to be produced with a good bokeh, whereas many Japanese components, also made by the main producers, are not considered to be valid. Canon lenses seem to be an exception.

In general, highlights are more sensitive to bokeh, also because perfectly circular highlights show that the photo was probably taken with a completely open aperture; if instead they are polygonal, it means that the aperture was closed at least by one unit. Sometimes one can even count the number of blades which make up the aperture. The aperture therefore has an effect on the bokeh effect, as does the number of blades which are part of it.



■ **Figure 6.52**
Bokeh effect used with different aperture settings. Notice how by decreasing the depth of field, highlights become more and more circular.

PHOTOGRAPHIC CAMERA

Until now, we have talked about photography, meaning the results which can be obtained by means of traditional, digital, compact or reflex cameras. In actual fact there are two other types of devices used for acquiring images and namely the movie camera and the digital video camera.

MOVIE CAMERA

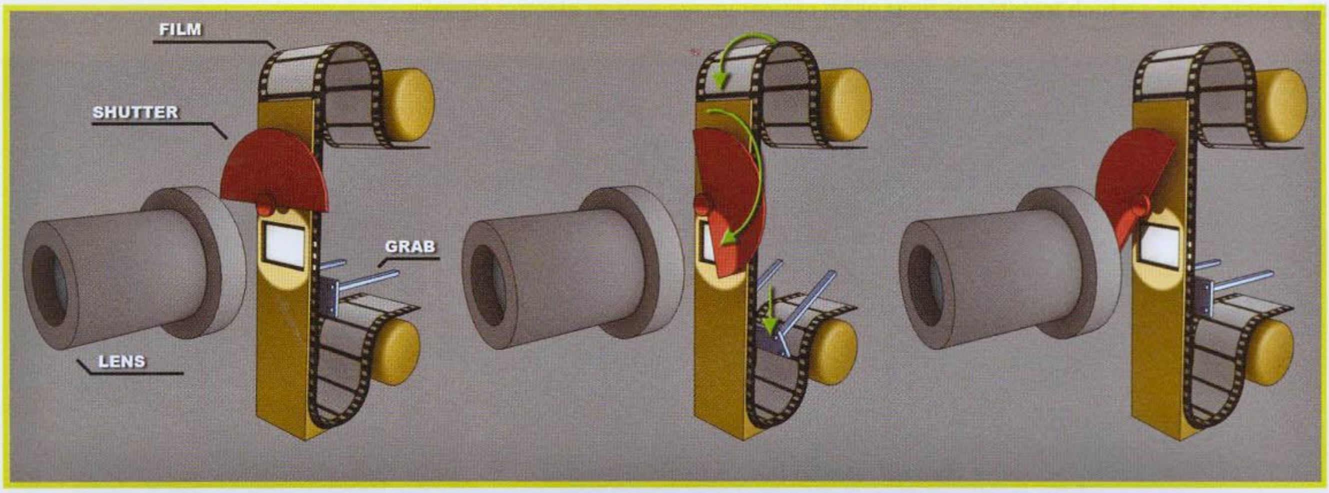
A video camera is an appliance which impresses a sequence of pohotographic images on a continuous film. Unlike the camera, which takes one picture at a time, a video camera takes a series of pictures which can then be presented as animations by means of a projector.

■ **Figure 6.53**
A traditional movie camera. Notice the enormous block inside which the film is placed.

The photosensitive surface is a film, which is made up of thousands of images called frames, which are placed in succession. In each frame the subject is in a slightly different position compared to the previous one. If these frames are watched at the right speed, one perceives a feeling of continuous movement. This is caused by a phenomenon called "vision persistence", which is the capacity of the eye to maintain the image for a fraction of a second also after the image is no longer actually visible. The speed of acquirement varies between 25 and 30 frames for each second of filming.

■ **Figure 6.54**
A cinema reel and its 35mm film.

In order to obtain this performance, the movie camera must carry out very fast and precise movements. The film cannot simply run through the back of the objective, or images would be blurred. In order to avoid this, the film, or rather a the frame, is placed behind the objective, blocked and exposed to light. Then the film is made to move forward to the next frame to be exposed. Like in a camera, also in this case it is the duty of the shutter (called rotating shutter) to define the amount and time span of exposure. When it is open, the film stops and is exposed for the amount of time decided by the user (or the machine if it is in automatic mode). Once closed, the film is moved on. This movement is caused by a small tooth which slips into the holes on the side of the film and drags it forwards. Once the film has been repositioned, the tooth withdraws and the new frame is exposed. This happens 24 times per second.



■ **Figure 6.55**
Scheme of how a movie camera functions.

The typical feature of this machine is the rotating shutter. This nothing more than a disc placed in front of the objective of a movie camera and made to rotate at a constant speed by means of a small battery-driven engine, alternating moments of darkness to moments of exposure.
In the previous image the shutter is formed by a fixed semi-circular object of 180°, but in truth this angle can be changed by the operator.

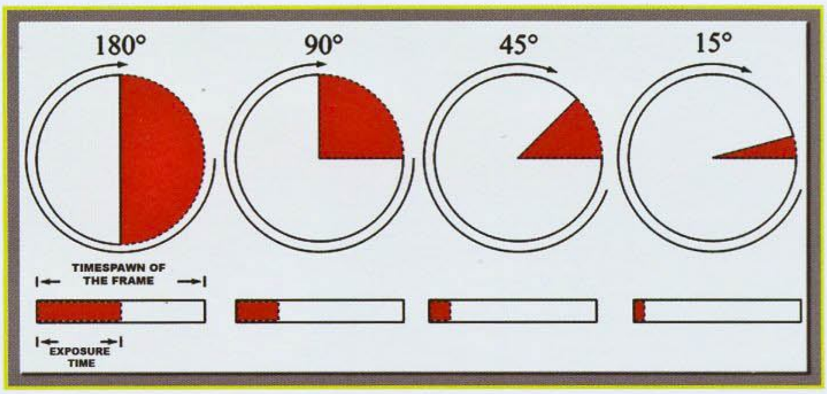
In order to calculate the exposure time of the film one can use the following formula:

$$\frac{\textit{Shutter angle}}{360 * fps} = \textit{Exposure measured in seconds}$$

where:

Shutter angle: the opening in degrees of the shutter.
fps: speed of the film.
360: this is a constant, representing the round angle.

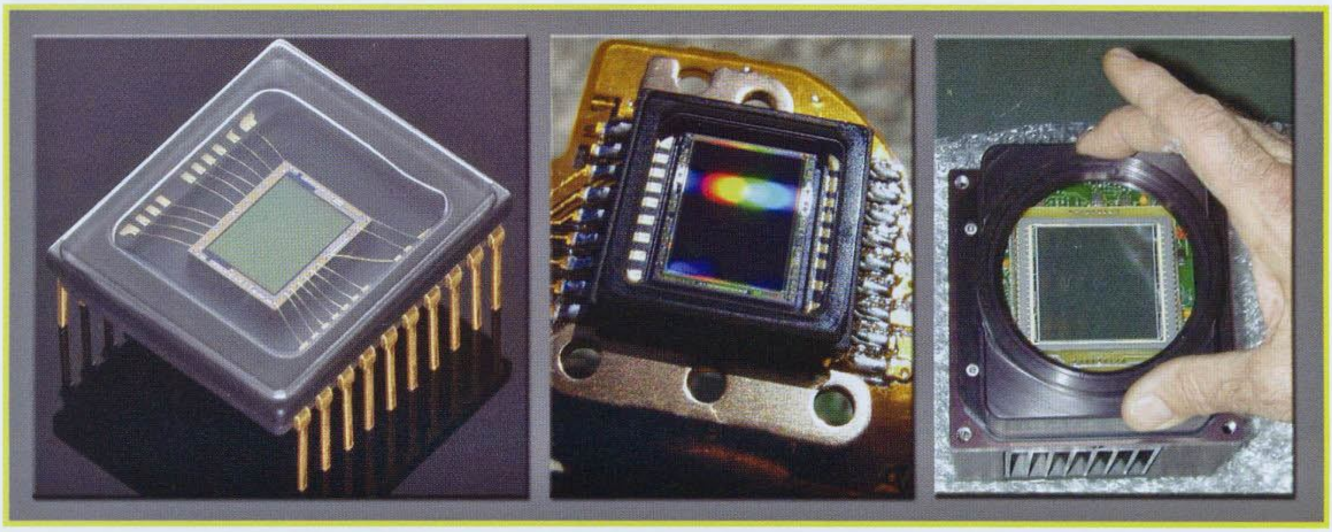
Following the previous example, where the opening is 180°, by using an fps of 24, one achieves a time of 1/48 sec. Changing the degrees of shutter opening causes a greater or lesser incidence of light on the film surface, as one would expect, similarly to what happens with cameras.



■ **Figure 6.56**
Various settings of the opening angle of the shutter.

DIGITAL VIDEOCAMERA

As far as the digital videocamera is concerned, things are slightly different. Instead of film, there is a CCD electronic sensor (**C**harge **C**oupled **D**evice). This element consists of a circuit formed by a grid of semiconductors able to accumulate an electric charge which is proportional to the intensity of electromagnetic radiation which hits them. By sending a series of impulses to the photographic device, one has an electric signal as an output, thanks to which it is possible to reconstruct the matrix of pixels which make up the image projected on the surface of the CCD itself. This information can be used directly in its analogue form, in order to reproduce the image on a monitor or record it on magnetic devices, or it can be converted to digital format to be saved as a file and later used again. Basically, it is an electronic device which transforms light or an optical image into an electric signal.



■ **Figure 6.57**
Some CCD sensors. This element was born at the Bell laboratories of Murray Hill, New Jersey, already the birthplace of the transistor in late 1969.

As well as the important CCD and other more common optical components, the digital videocamera contains a rather particular shutter, an electronic shutter. This works by activating and deactivating the CCD so that it does not record light, even if it is hit by light. The shutter time is thus controlled, and can go from thousandths of a second to a few minutes or more.

VRay Camera : VRayPhysicalCamera

INTRODUCTION

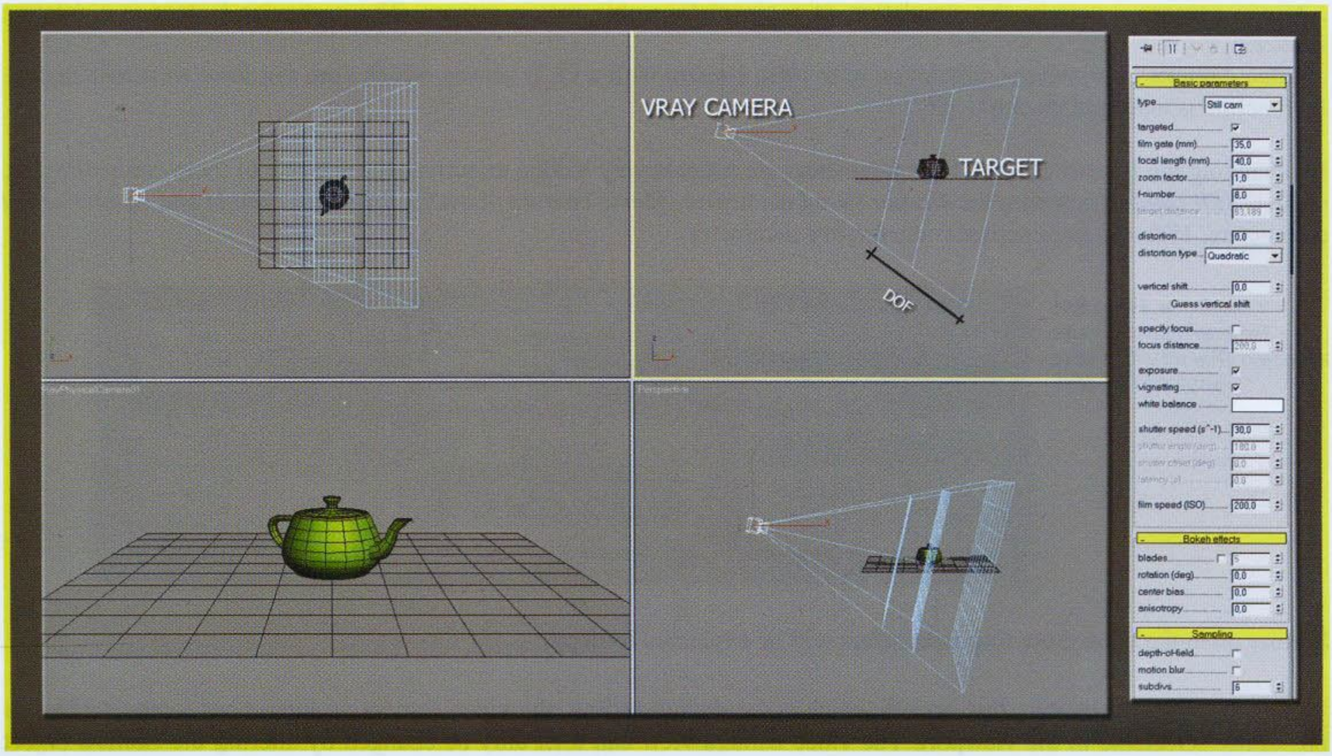
In the previous pages the main concepts concerning photography and cameras in general have been illustrated.
VRay, thanks to a special virtual camera installed directly within 3ds Max, allows one to recreate and use real parameters. This goes by the name of ***VRayPhysicalCamera***, which is a camera with physical values. It is therefore possible to input settings such as aperture or film sensitivity. This allows the user, with adequate preparation, to set renders for very realistic and physically correct renders, because the parameters used are the same ones found in real life.

PARAMETERS

The ***VRayPhysicalCamera*** is located in the Command Panel of 3ds Max, in the Create tab, together with the 3ds Max cameras. *VRay* offers two types of cameras, where the most complete definitely the ***VRayPhysicalCamera***.

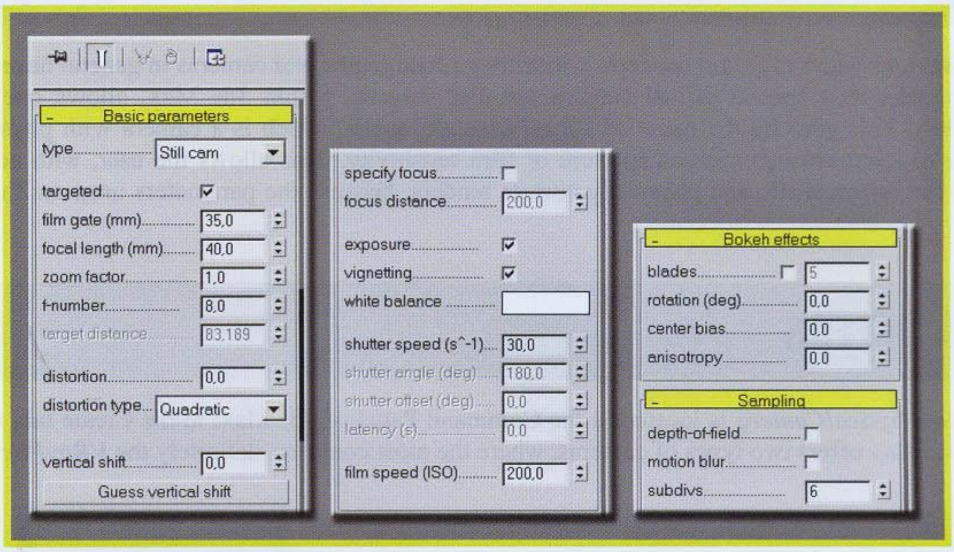
■ **Figure 6.58**
The menu where one can select the two different cameras.

For its creation, the procedure is identical to the one used for creating the 3ds Max video cameras.



■ **Figure 6.59**
The *VRayPhysicalCamera* with its parameters.

What is generated is an object which is only visible through the viewport, similar in every way to a camera, with the machine body, represented by a small box, and a ***target*** (the camera's point of view) which can both be selected by the user. All of this is connected with the visual cone, which has three levels. The ones closest and farthest from the machine body represent the limits of the depth of field, instead the central one indicates the point of perfect focus of the objective.
The modification parameters are divided into three rollouts, of which the first, ***Basic parameters***, is the most complex and contains the most important control parameters. The remaining two, ***Bokeh effects*** and ***Sampling***, allow the modification of effects such as ***DOF***, ***motion blur*** and ***Bokeh***.



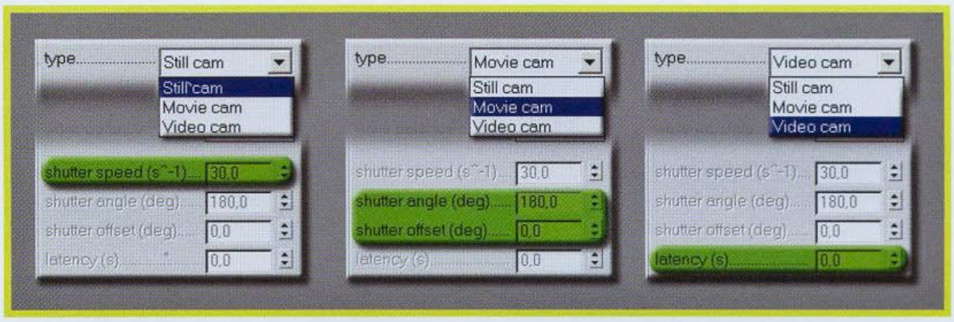
■ Figure 6.60
The modification parameters of the ***VRayPhysicalCamera***.

Basic parameters

type - this pull-down menu allows one to select three different types of camera. The selection of one of these directly influences the type of ***motion blur*** produced.

. Still camera - permits the simulation of the classic camera with film. This indicates the presence of a mechanical shutter, an aperture and an objective.
. Movie camera - permits the simulation of a cinematographic movie camera. The particular feature is a rotating shutter.
. Video camera - permits the simulation of a video camera with a CCD sensor, which does not have an actual shutter. Its function is managed electronically.

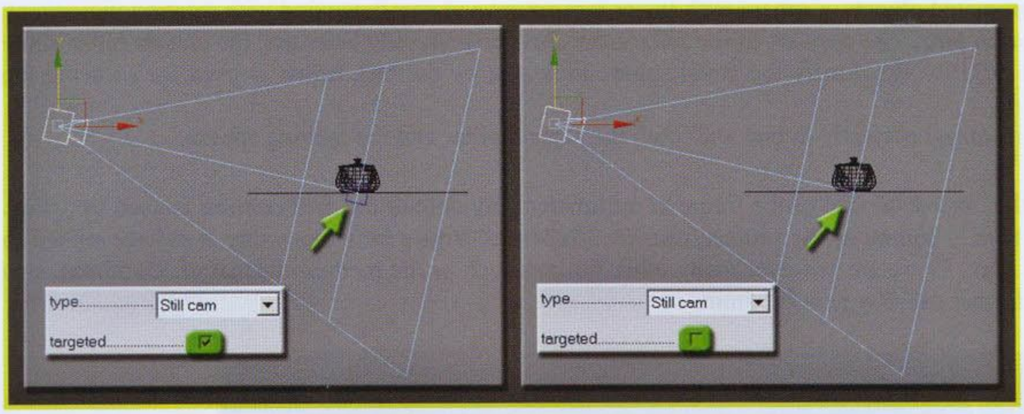
By changing the type of camera, four parameters are alternatively hidden or activated, each one of them concerning the shutter. ***Still camera*** activates the ***shutter speed*** parameter, ***Movie camera*** activates the ***shutter angle*** and ***shutter offset*** parameters, ***Video camera*** activates the ***latency*** parameter.



■ Figure 6.61
The three types of cameras available in VRay with their relative parameters activated.

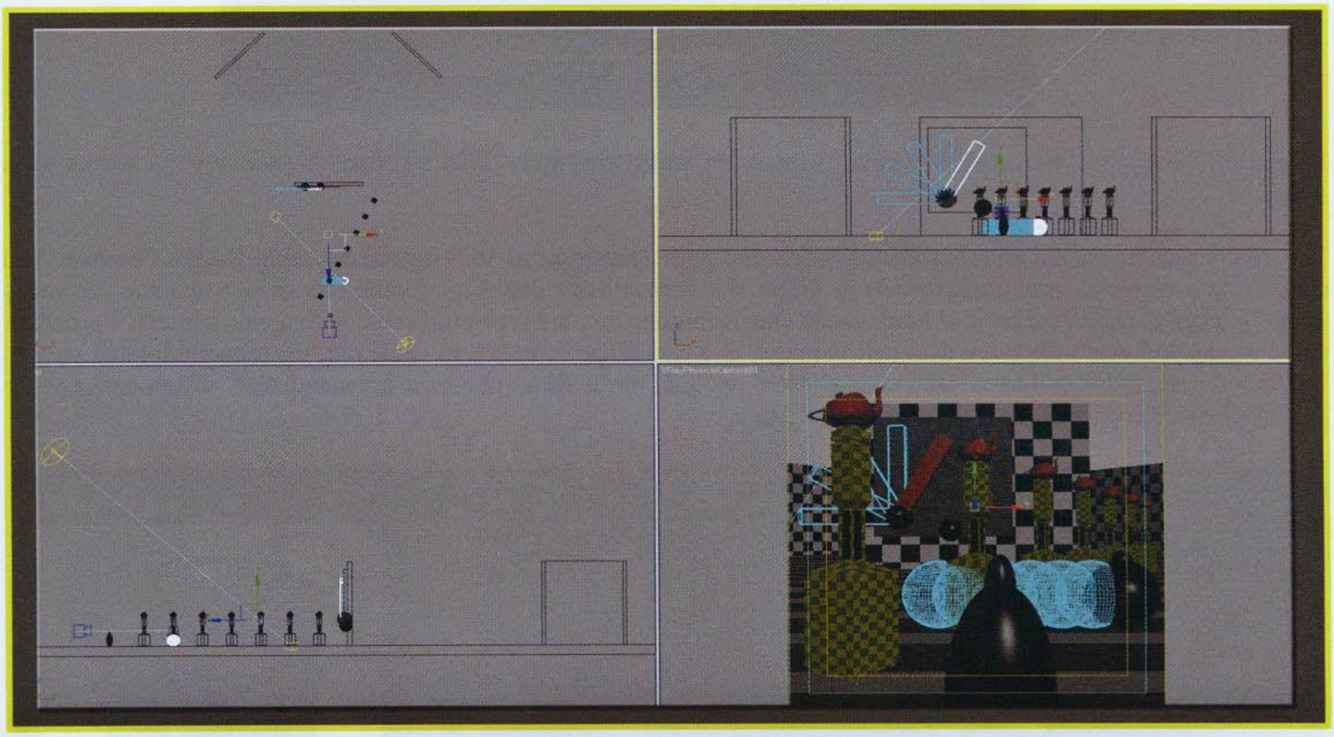
Further on in the book these four parameters will be explained in-depth.

targeted - by deactivating this option, which is the default setting, the video camera no longer has a ***target*** which is physically selectable in the viewport. The video camera view can then only be changed by altering the ***camera*** object and not the ***target*** object, which instead is what happens if this option is activated.



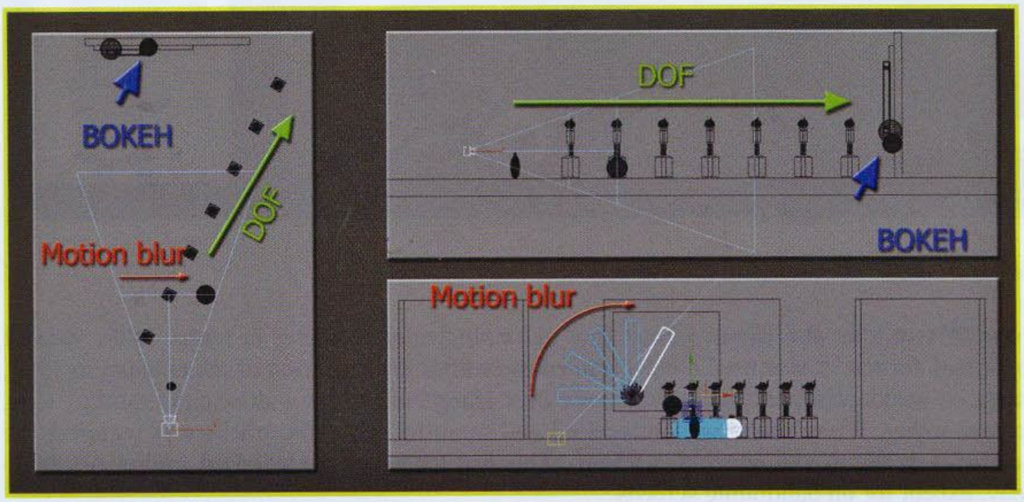
■ **Figure 6.62**
By deactivating the ***targeted*** parameter, the small box which allows one to direct the camera view disappears. By doing this, in order to change the view, one must rotate the camera object.

Before moving on to explain the parameters of the ***VRayPhysicalCamera***, it is necessary to give a brief explanation of the scene used for the following texts.



■ **Figure 6.63**
Images of the scene being examined.

It is a scene made up of simple geometrical objects, which have been placed in such a way to exploit the ***VRayPhysicalCamera*** and its parameters to their full potential.

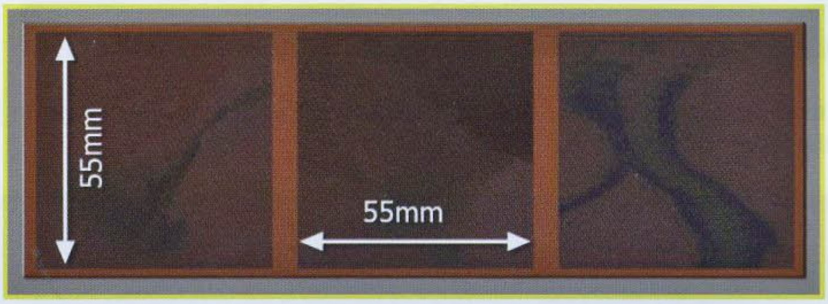


■ **Figure 6.64**
The choice and position of the objects has been thought out so that the effects of the ***VRayPhysicalCamera*** are emphasized.

In this scene there are two animated objects, a sphere with linear movement and parallelepiped with rotating movement. This is useful for testing ***motion blur***. The position of the pillars, the checker-textured planes and the closest pillar allow one to observe the way the ***DOF*** works. Lastly the small sphere indicated by the blue arrow is used for showing the ***Bokeh*** effect.
Notice how the camera has its ***target*** perfectly in line with both the second pillar and the moving sphere.

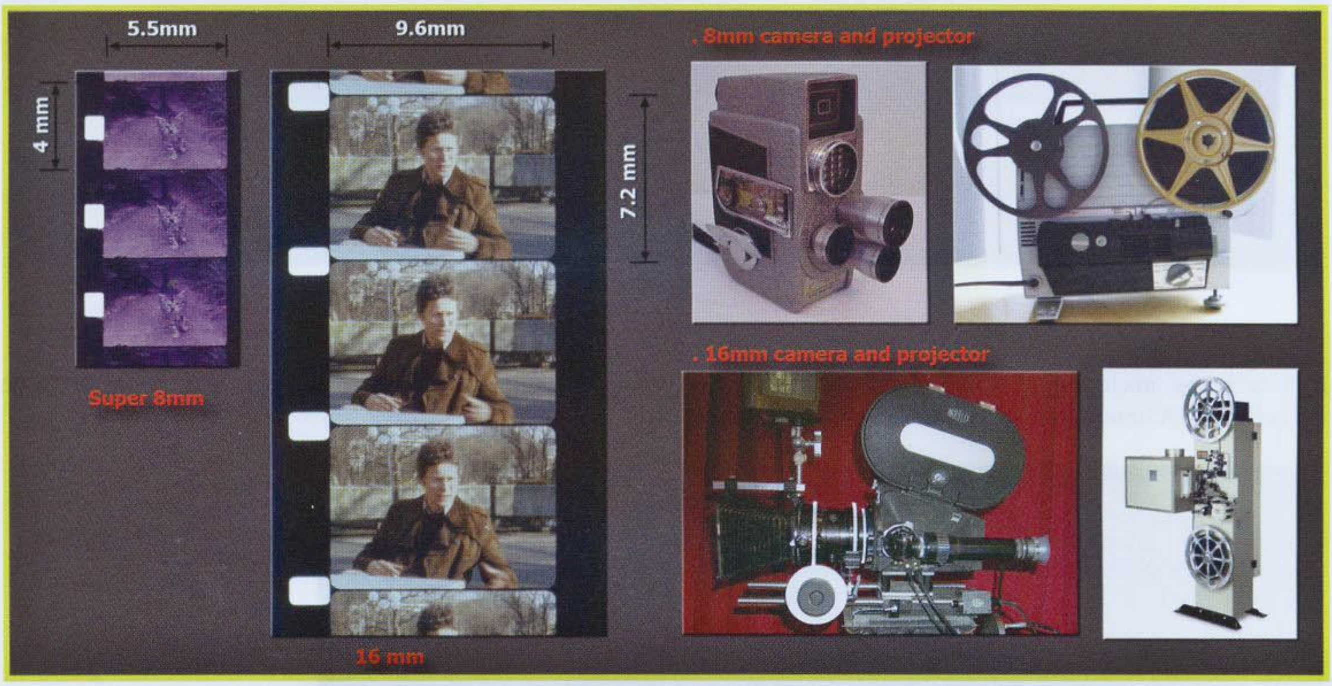
film gate - this specifies the horizontal size of the frame in millimeters. By default the ***Still camera*** is used by ***VRay***. The size of a photographic film is 36mm, as the frames measure 36x24mm. Notice how this value is strictly related to the System Unit within 3ds Max. There are however many other formats such as the medium format of 55x55mm, also for photo cameras. For this format the ***film gate*** is 55mm.

■ Figure 6.65
The medium 120 format of 55mm.



Instead for the cinema, the smallest format is the super 8mm, used since 1965 by amateur operators. Its size is 5.4 x 4mm.

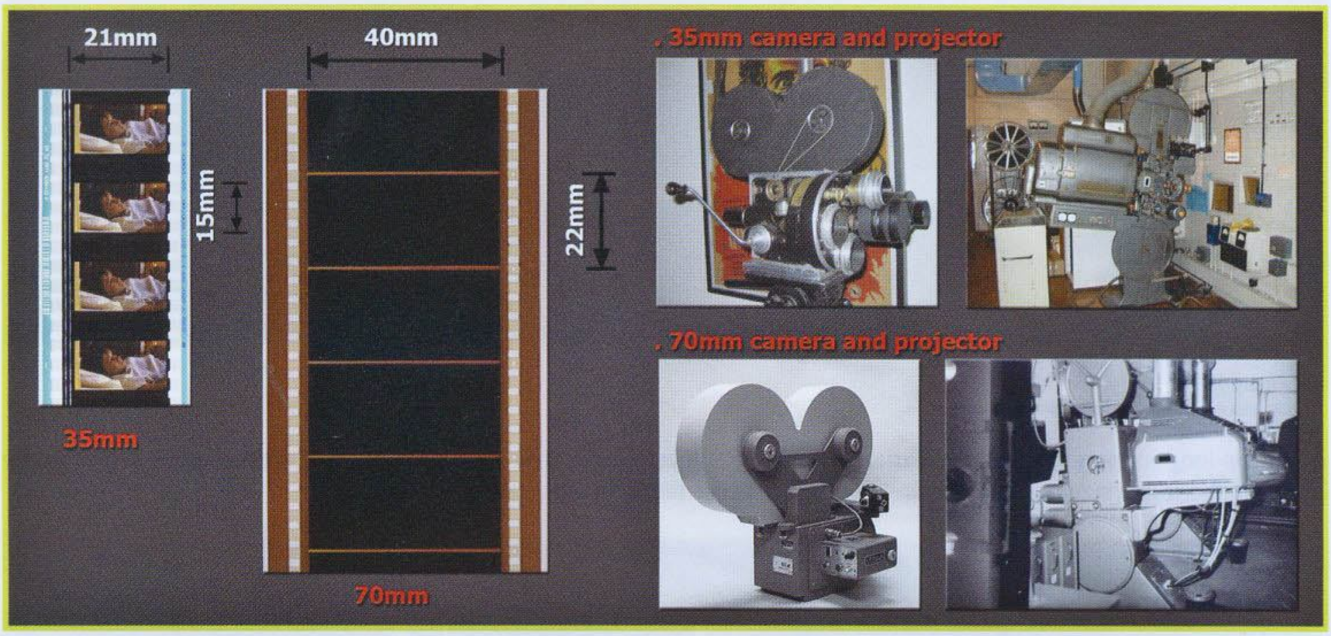
The 16mm cinema format was introduced before the 8mm, precisely in 1923 by the Eastman Kodak Company and destined to non-professional amateurs. At its origin, this format had a double perforation on its side and was obviously mute. In order to tackle the field of home-made cinema production, the set would come with a cine-camera, a projector and a tripod, and cost little more than half the price of a car. When soundtracks came out, the monoperforated version was introduced, and is still used today: this allowed one side of the film to be used for soundtrack, at first optical, then also magnetic. The size of the film is 9.6mm x 7.2mm.



■ Figure 6.66
The two 8mm and 16mm formats compared.

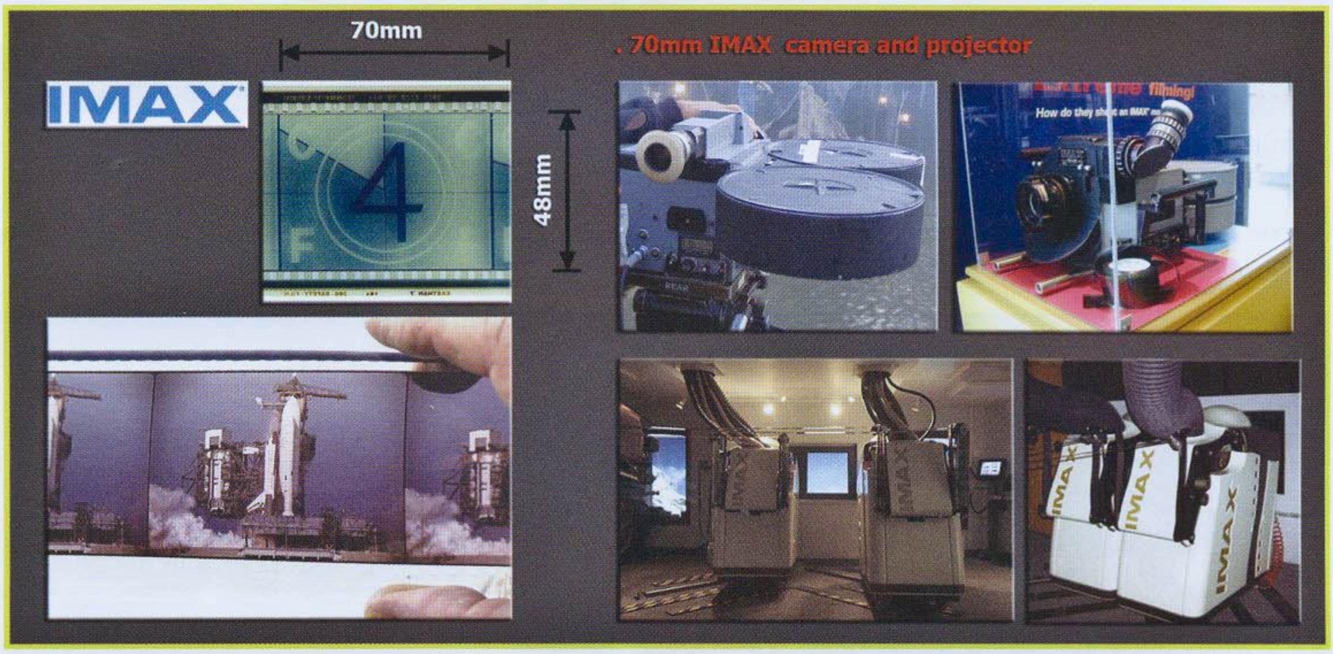
The following format in terms of size after the 16mm format (not chronologically speaking in terms of the date of introduction) was the 35mm cinema format. It was used almost immediately after the invention of the cinema, in 1909, after an international agreement on standardization. Each frame was 24 x 18mm and had a width-height ratio of 1.33, like that of 4/3 television screens today. When sound was introduced and room for the audio tracks was required, the frames were initially changed to 21 x 18mm and later, 21 x 15mm. This last format is the standard, called Academy Standard and is still used today, except for on panoramic screens.

The 35mm format was the starting point for all the other formats, which inherited its main characteristics, with the introduction of very few new variations: as we have already said, it is still used nowadays and is in fact the standard projection format used by all professional projectors in the world.
Moving on, we find the 70mm format. This film allows a larger frame to be captured compared to the 35mm and this gives images a superior definition and a more vivid quality. The 70mm has a size of 22 x 48mm. This means that the quality of images has improved three times as much compared to projections carried out with a normal 35mm film.



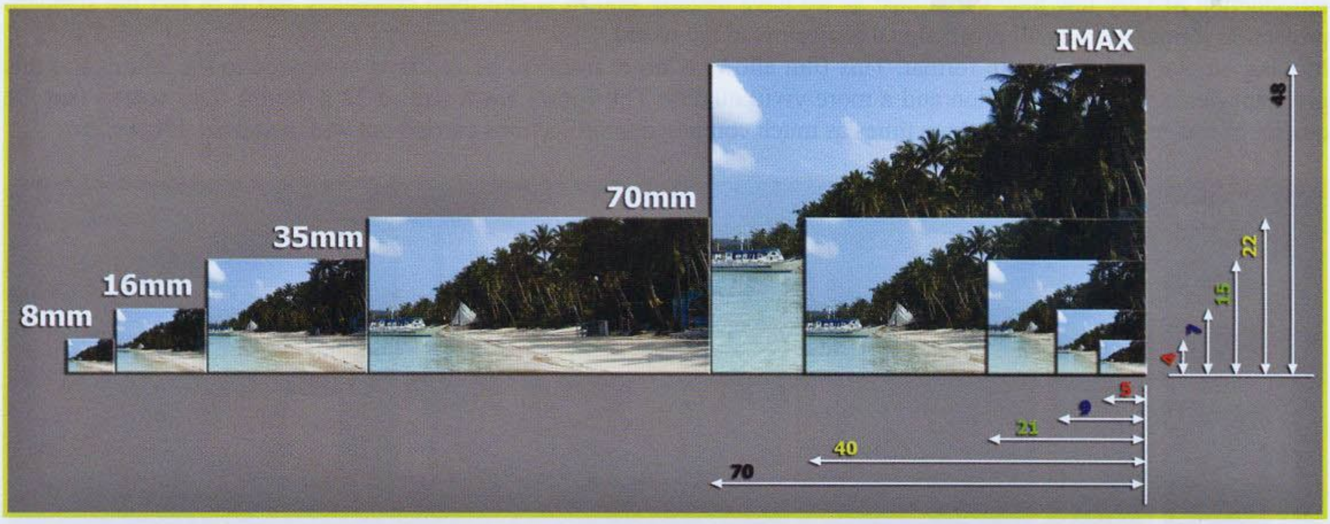
■ **Figure 6.67**
35mm e 70mm film format.

The IMAX was the greatest film format in the history of the cinema, with a size of 70x48mm, 10 times as large as the current standard of 35mm and 3 times as large as the 70mm. In order to expose the film at the standard speed of 24 frames a second, the film must run at a speed which is three times as fast as the standard 35mm.



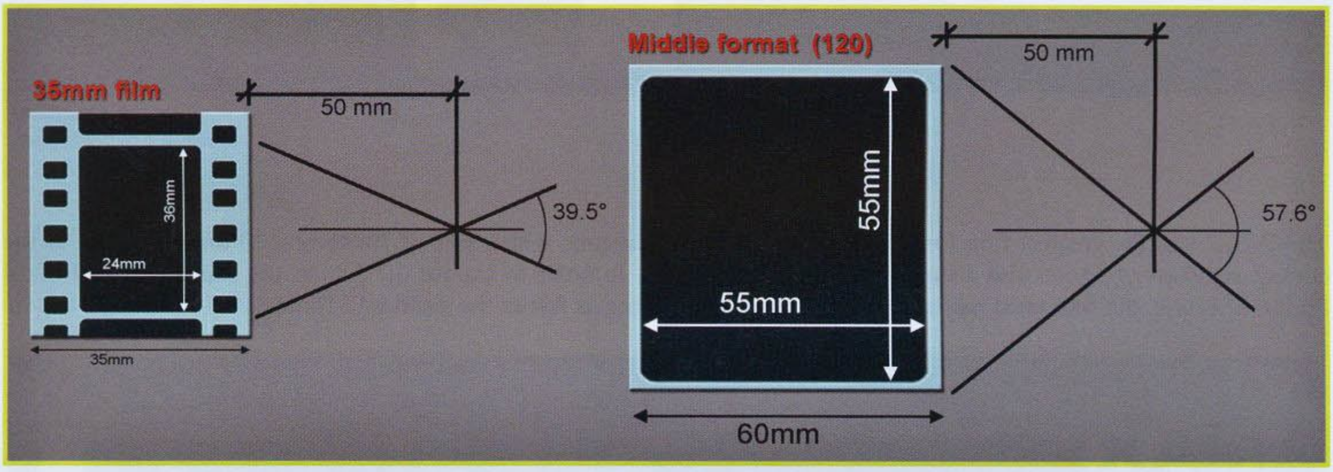
■ **Figure 6.68**
The IMAX format in all its glory!

Lastly, here is a graphic summary of the most used cinema formats.



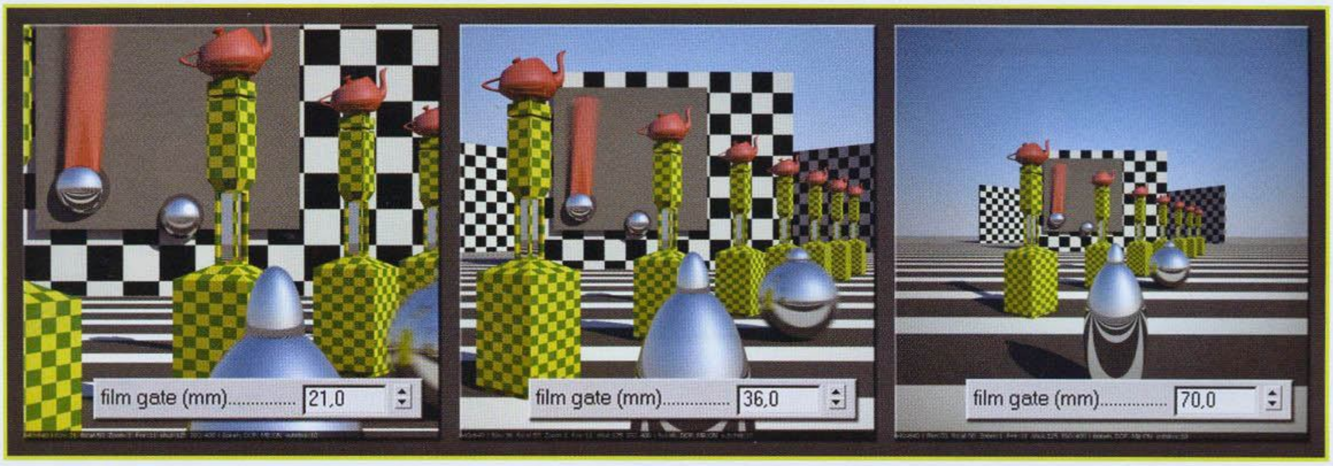
■ **Figure 6.69**
Comparison between various cinema formats.

After this long journey through the variety of photographic formats, it is appropriate to ask what actually happens by changing the frame size or the ***film gate***. For this, observe the following image:



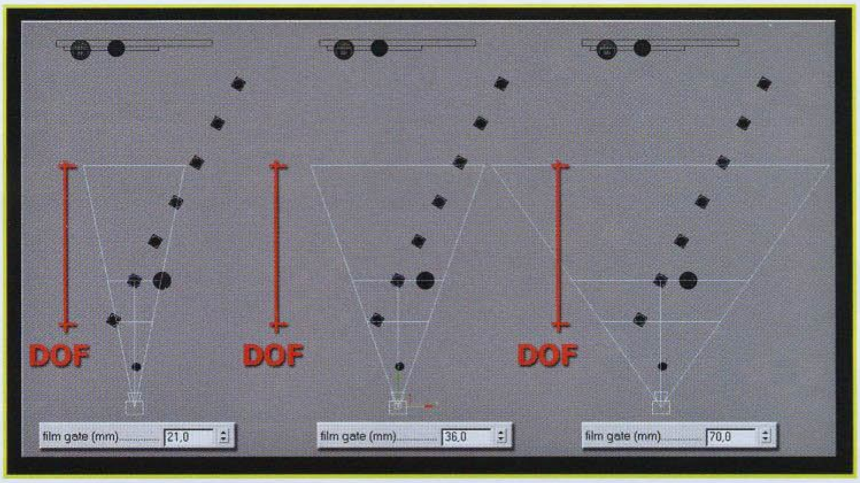
■ **Figure 6.70**
Different types of film allow different angles of view.

As one can see, by keeping the same objective, in our case the 50mm, and only changing the type of film, one can capture a wider field of view, thanks to a greater angle of view.



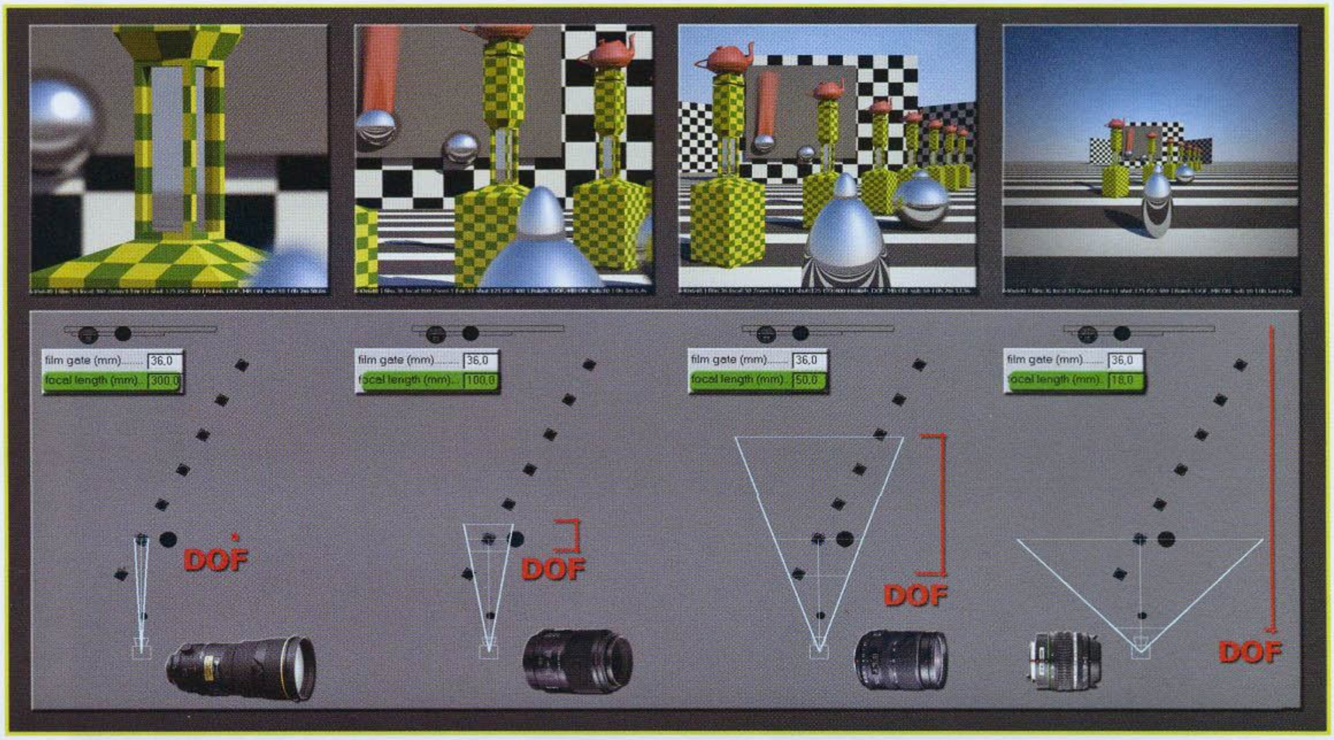
■ **Figure 6.71**
Different ***film gate*** values. Notice the variation of the field view.

By observing the behavior of the *VRayPhysicalCamera* in the viewport, one finds that the depth of field is not influenced by film size.



■ **Figure 6.72**
Variation of the ***film gate*** in reference to the ***DOF***.

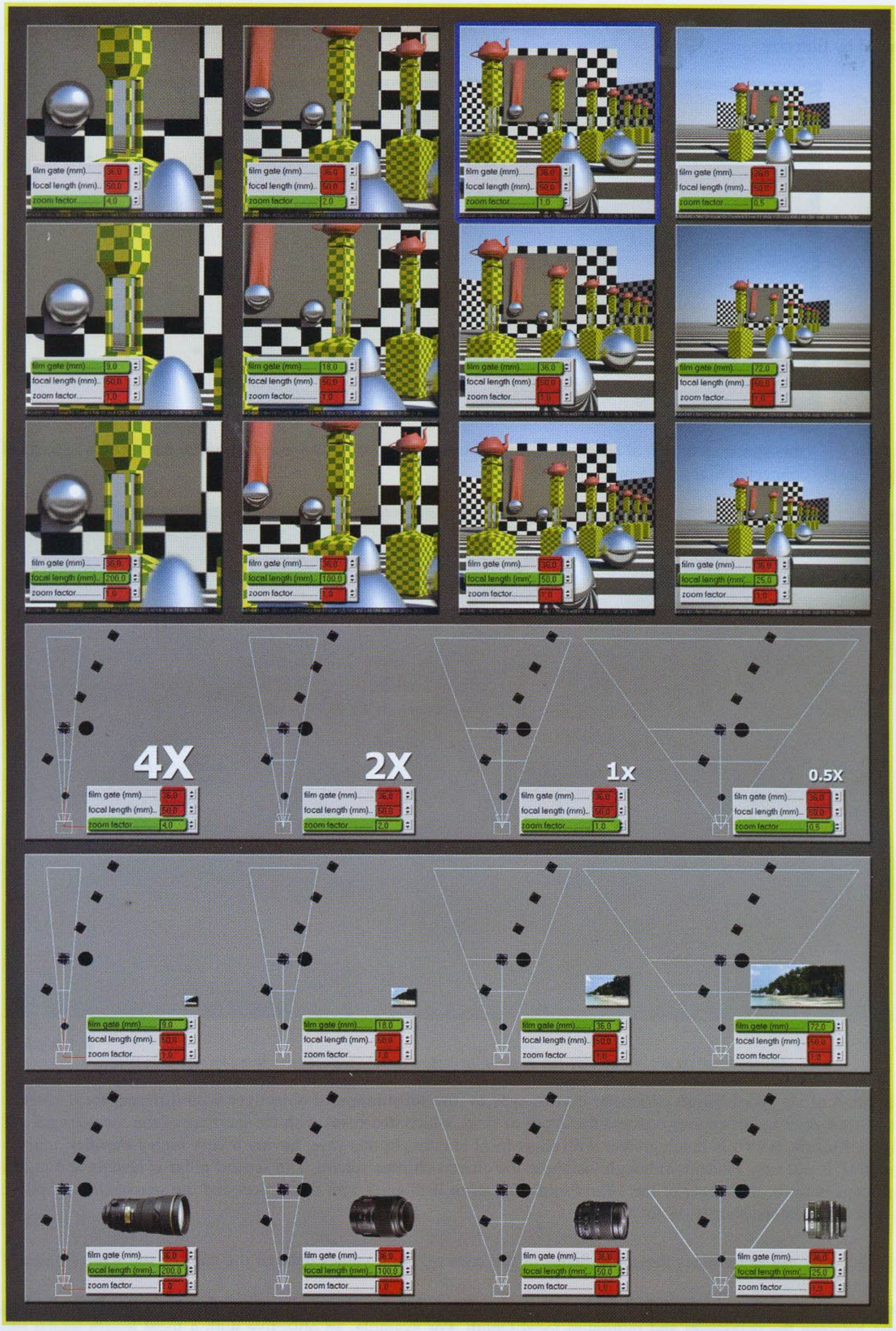
focal length - this specifies the ***focal length*** of the objective. Similarly to previous examples, also in this case focal length is closely connected to the System Unit in 3ds Max.



■ **Figure 6.73**
The variation of the type of focal length directly affects the depth of field and angle of vision. Notice the ***vignetting*** effect in the last render.

The variation of the ***focal length*** parameter corresponds to the substitution of objectives with different focal lengths. The most important modifications concern the depth of field, which decreases with the increase of the focal length and the angle of vision, which is much greater with short focal lengths. In this case, having a very high ***f-number***, namely 11, the ***DOF*** is not very evident, although visible in the closest chrome sphere. The second pillar is always and in any case in focus because the ***target*** of the ***VRayPhysicalCamera*** is aligned with that object and the ***specify focus*** option, which will be explained shortly, is not active.

zoom factor - this determines the ***zoom*** factor of the camera. High values enlarge the subject and values below 1 make it smaller.
A ***zoom*** is a complex objective and its focal length can be changed by the user. Often one refers to a ***zoom*** by indicating the ratio between the maximum and minimum focal length. For example, an objective with a focal length which can range from 100mm to 400mm could be indicated as a "4x zoom". Sometimes the term hyperzoom is used to define a ***zoom*** with a very big extension, usually between "5x" and "10x", like for instance from 35mm to 350mm or even a "12x". Observe the following image to understand the connection between the parameters better.



■ **Figure 6.74**
Modification of the ***zoom factor***, ***film gate*** and ***focal length*** parameters. The result is always the same angle of vision.

In the previous image, the same visual angles have been obtained by each time changing a different parameter amongst ***film gate***, ***focal length*** and ***zoom***.
In the test we began with reasonably standard parameters, such as those set in the render highlighted in blue. First of all we modified the ***zoom*** parameter with exponential values of 2, that is 0.5, 1, 2 and 4. What we have obtained is a change in the angle of view. The ***DOF*** instead has not changed. The exact same result can be had by blocking the ***zoom*** value and the ***focal length*** and changing the ***film gate*** parameter: the renders obtained this way are identical to the previous ones and depth of field is unaltered in all cases. There is however a change in the ***vignetting*** effect. By using the ***zoom*** and not ***film gate***, ***vignetting*** is practically absent. Instead by changing focal length, both the view angle and the depth of field vary. In this case one can obtain the same field of view as the previous ones, but the ***DOF*** has been changed this time.

FIELD OF VIEW	10.2°	20.4°	39.5°	71.5°
Film gate	36mm	36mm	36mm	36mm
Focal lenght	50mm	50mm	50mm	50mm
Zoom	4	2	1	0.5
Film gate	9mm	18mm	36mm	72mm
Focal lenght	50mm	50mm	50mm	50mm
Zoom	1x	1x	1x	1x
Film gate	36mm	36mm	36mm	36mm
Focal lenght	200mm	100mm	50mm	25mm
Zoom	1x	1x	1x	1x

■ **Figure 6.75**
Some combinations of the ***film gate***, ***focal length*** and ***zoom factor*** parameters. To equal colors, equal view angles correspond.

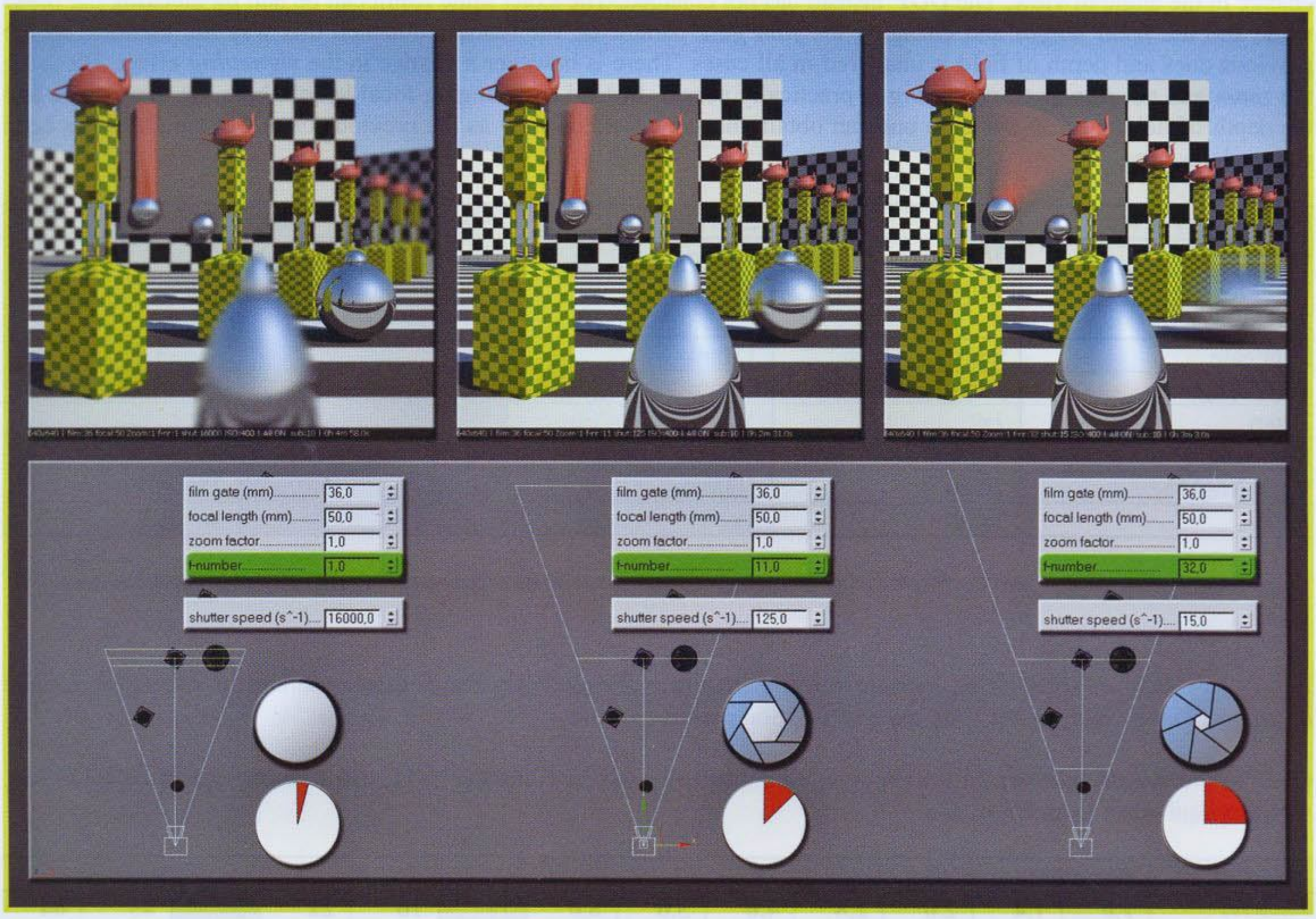
f-number - this establishes the aperture opening amount of the machine. The ***f-number*** parameter affects the amount of exposure in the image only if the ***exposure*** option is active.

Observe the following chart. In the previous examples we used an ***f-number*** = 11 and a ***shutter speed*** of 1/125 sec. which is equivalent to an EV of 14.

	f-number												
Time (s)	**1.0**	**1.4**	**2.0**	**2.8**	**4.0**	**5.6**	**8.0**	**11**	**16**	**22**	**32**	**45**	**64**
60	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
30	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
15	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8
8	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
4	-2	-1	0	1	2	3	4	5	6	7	8	9	10
2	-1	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	3	4	5	6	7	8	9	10	11	12
1/2	1	2	3	4	5	6	7	8	9	10	11	12	13
1/4	2	3	4	5	6	7	8	9	10	11	12	13	14
1/8	3	4	5	6	7	8	9	10	11	12	13	14	15
1/15	4	5	6	7	8	9	10	11	12	13	14	15	16
1/30	5	6	7	8	9	10	11	12	13	14	15	16	17
1/60	6	7	8	9	10	11	12	13	14	15	16	17	18
1/125	7	8	9	10	11	12	13	14	15	16	17	18	19
1/250	8	9	10	11	12	13	14	15	16	17	18	19	20
1/500	9	10	11	12	13	14	15	16	17	18	19	20	21
1/1000	10	11	12	13	14	15	16	17	18	19	20	21	22
1/2000	11	12	13	14	15	16	17	18	19	20	21	22	23
1/4000	12	13	14	15	16	17	18	19	20	21	22	23	24
1/8000	13	14	15	16	17	18	19	20	21	22	23	24	25
1/16000	14	15	16	17	18	19	20	21	22	23	24	25	26

■ **Figure 6.76**
Chart with equivalent EV values.

Now imagine changing the ***f-number***. By using a less open aperture, in order to obtain the same amount of exposure as before, one must expose the film for longer. It is important to remember that the size of the film plays an extremely important role, both regarding the ***DOF*** and indirectly also the ***motion blur*** effect. The more closed the aperture is, the greater the depth of field. As a result, exposure time must be longer, and therefore ***motion blur*** is favored.

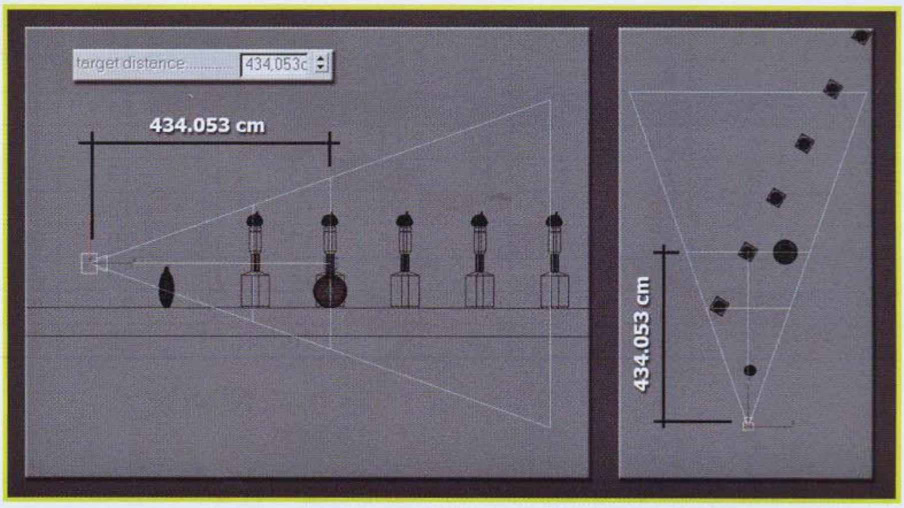


■ **Figure 6.77**
Variation of the ***f-number*** parameter.

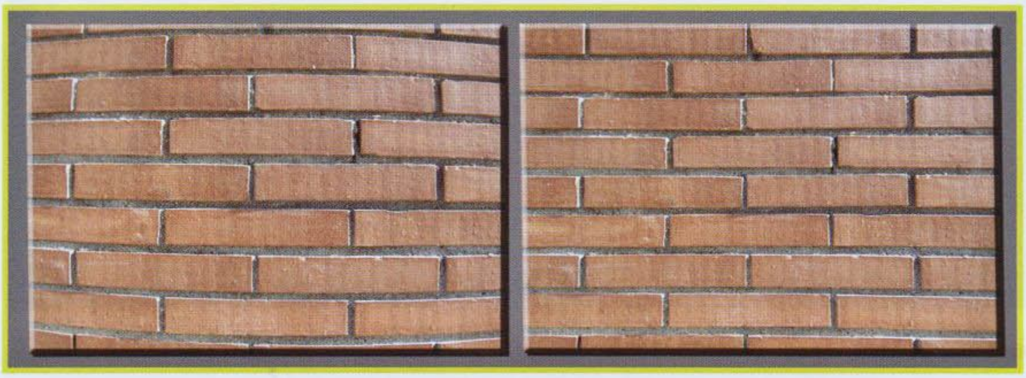
In the first image one can notice a low depth of field, with objects out of focus both lose-up and in the background. Also, the moving objects, such as the chrome sphere in line with the second pillar and the red pointer needle in the background do not have any ***motion blur*** effect.
A completely opposite situation appears in the last image. The ***DOF*** is practically absent, whereas the trails left by objects are quite evident. Notice the correction of exposure time for the compensation of different aperture settings.

target distance - it is an information parameter, which gives the distance between the camera and its ***target***.

■ **Figure 6.78**
The ***target distance*** parameter is a read only value, in fact it is grayed out and cannot be changed by the user. It shows the distance between the camera and its ***target***.

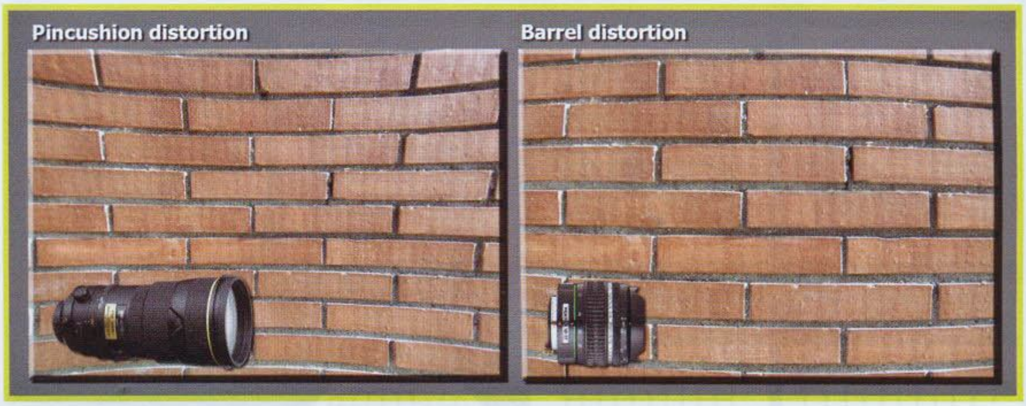


distortion - this specifies the amount of distortion applied to the lenses of the camera. One of the classic flaws of objectives which one considers at the time of purchase is distortion. Distortion can be evident when one takes a photograph of a subject which has a lot of straight lines, such a grid, a building, the sea on the horizon. In the case of a landscape, often distortion is not perceivable.
By taking a photograph of a wall, one would expect a result with rows of perfectly horizontal bricks. In actual fact this does not happen because of the distortion introduced by the objective of the camera.



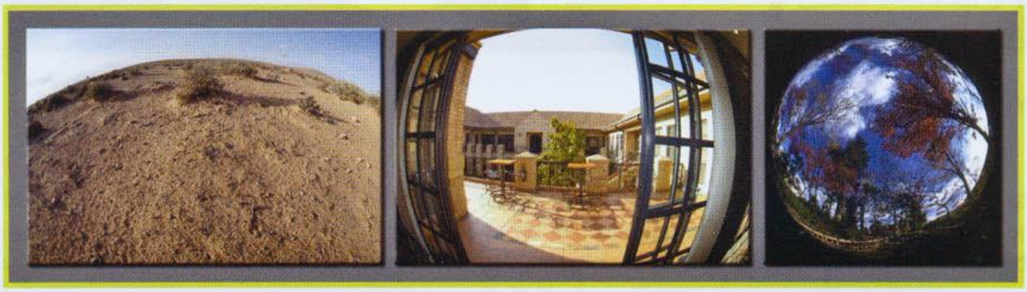
■ **Figure 6.79**
A comparison between an image with distortion and one without any. The distortion we can see in the right-hand image has been eliminated with Photoshop.

We are looking at barrel distortion; its cause is intuitable. This type of distortion mostly afflicts wide angle objectives and in particular wide angle zoom objectives and wide angle/tele zooms at the smallest focal length. Fixed objectives are often more correct.
If instead one uses teleobjectives with excursion, for example 28-70mm, it is common for a moderate barrel distortion to be present at minimum focal setting (28mm), whereas there is a perceivable pincushion distortion which is introduced in a completely different way.



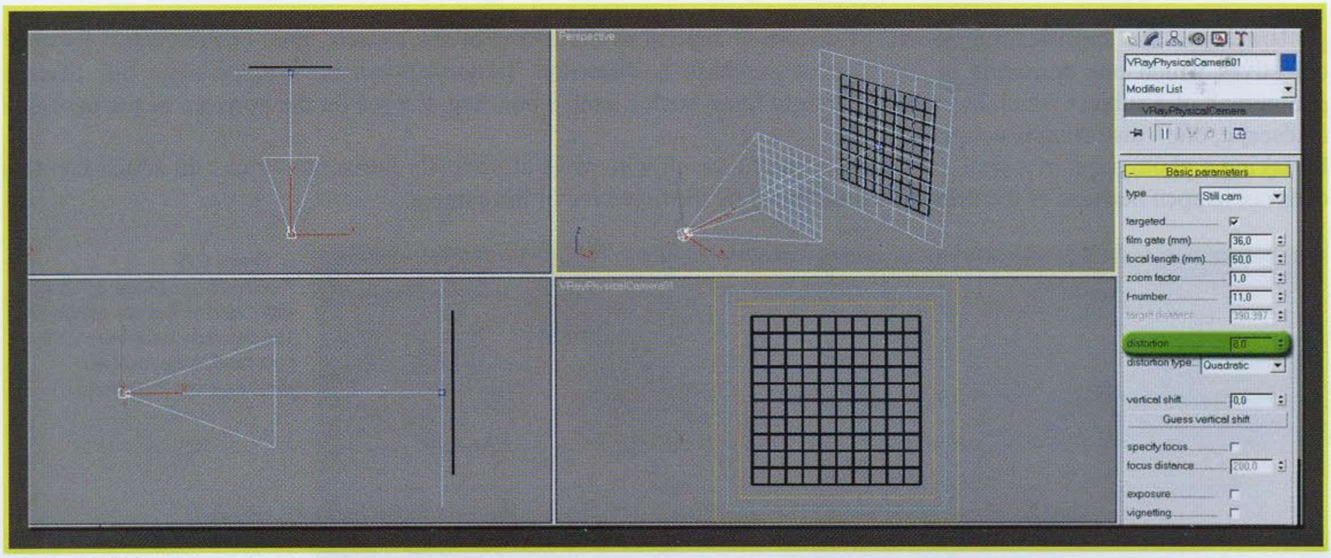
■ **Figure 6.80**
The two types of distortion compared.

Teleobjectives suffer distortion less than wide angles, whereas there are certain wide angles which make distortion their distinctive trait. This is the case of ***Fish-eye*** objectives which greatly emphasize barrel distortion.



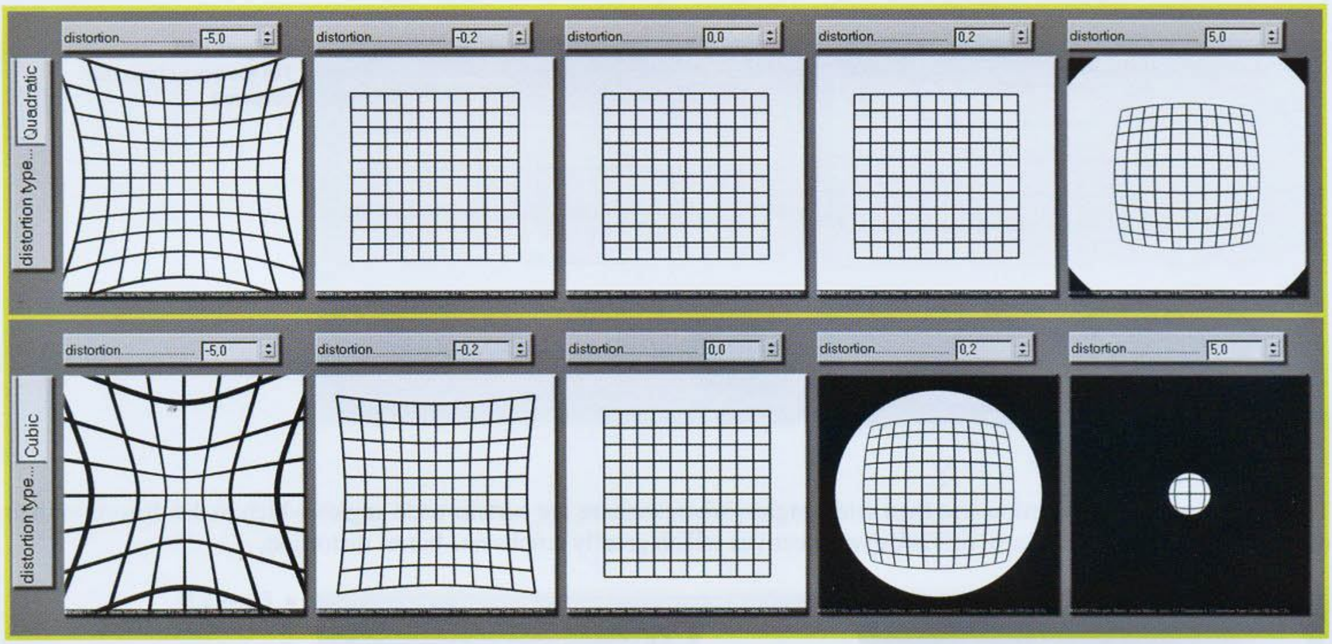
■ **Figure 6.81**
Some examples of barrel distortion.

In order to observe the way this parameter works, we have used a very simple scene, made up of geometric grid placed in a perpendicular position compared to the camera. The exposure options in this case are not active, because the ***exposure*** parameter has been deactivated.



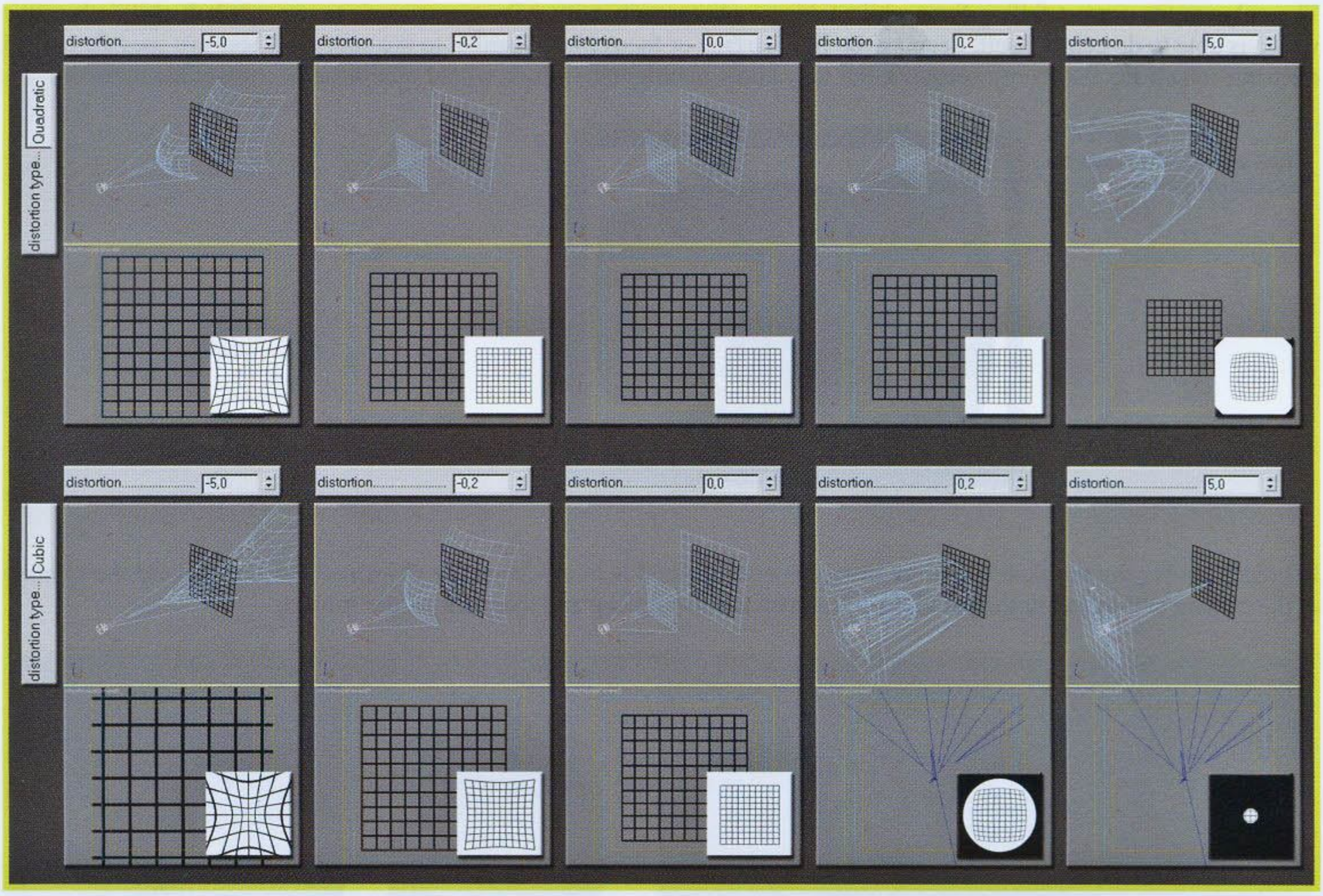
■ **Figure 6.82**
The scene used for explaining the ***distortion*** parameter.

By analyzing the following renders, with values lower than 0.0, pincushion distortion has been obtained. With values above 0.0, distortion is instead of the barrel type. *VRay* allows one to select two different types of distortion: quadratic and cubic. This last type has been introduced because it is used in programs like ***Syntheyes*** or ***Boujou***, and therefore allows a correct camera match with these products. Moreover, using the ***Cubic*** method quickens the drop in distortion. In the example with the values -0.2 and 0.2, the distortions created with the ***Cubic*** method are very evident compared to the ***Quadratic*** method, where distortion is missing.



■ **Figure 6.83**
The ***distortion*** parameter applied by using two different methods.

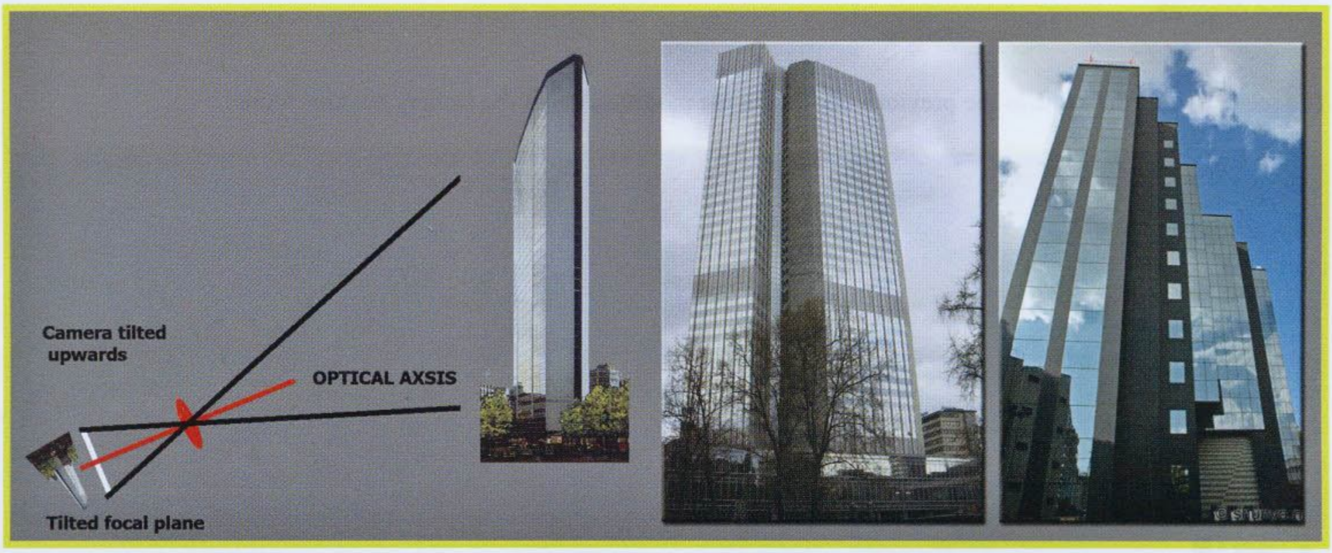
Changing this parameter has direct implications in the viewport, where both the shape of the camera and the visualization change with the variation of the ***distortion*** parameter. As far as real-time visualization is concerned, *VRay* does not curve geometries, but shows a preview of the size of the object once it has been deformed.



■ **Figure 6.84**
Variations in the shape of the camera and the visualization of the scene obtained with the ***VRayPhysicalCamera*** and the ***distortion*** parameter.

From the previous image one can immediately notice in the viewport that the difference between the two types of distortion is evident. One can also the curve in focal lengths due to the variation in optical distortion.

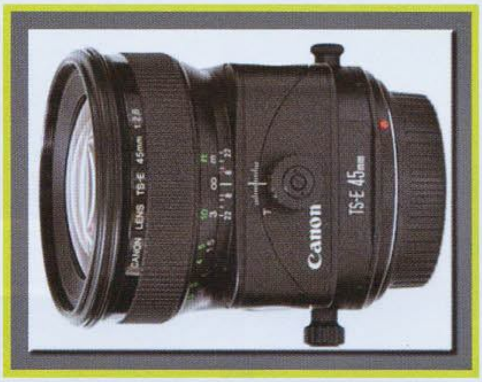
vertical shift - the distortion we have seen previously must not be confused with the phenomenon of "falling lines". With this term we mean the particular perspective effect which one obtains by tilting the camera upwards when taking a photograph of a building or a skyscraper. This problem is caused by the missing parallelism between the film plane and the plane defined by the vertical lines of the subject being photographed.



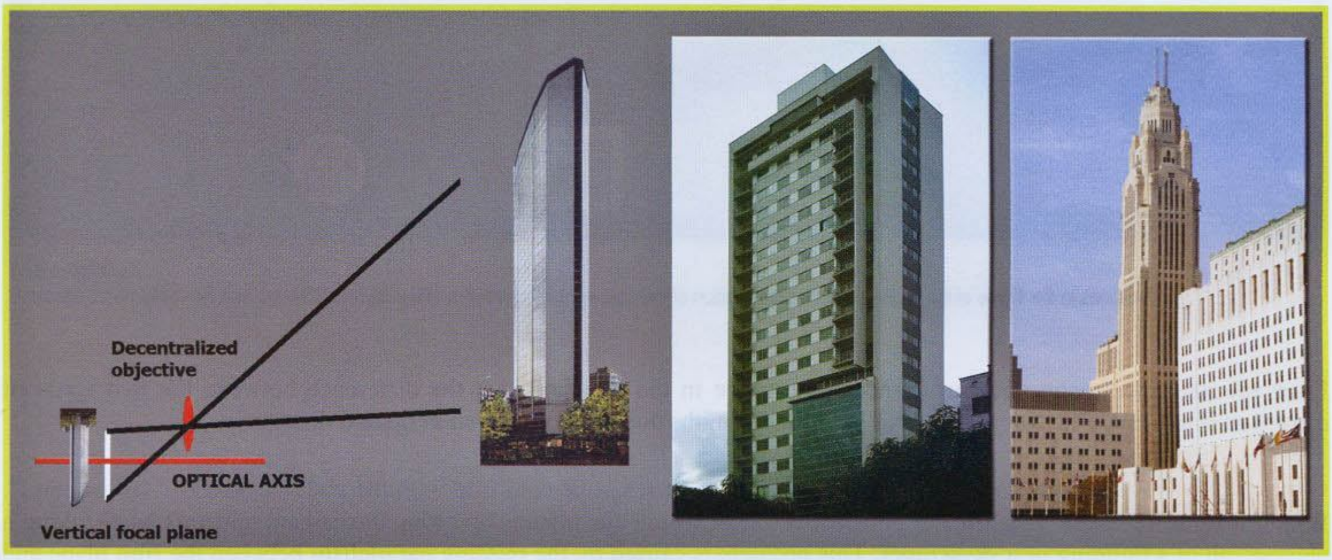
■ **Figure 6.85**
Examples of the problem of falling lines.

The only way to correct falling lines whilst taking a photograph, with reflex cameras, is to use a "tilt & shift" objective which is able to de-center itself, keeping the film surface or focus plane parallel to the subject. These are specific optical elements which are also very expensive, but they allow photographs which would otherwise be impossible to fulfill.

■ **Figure 6.86**
A ***tilt & shift*** objective. Notice the knobs for tilting and de-centering the objective.

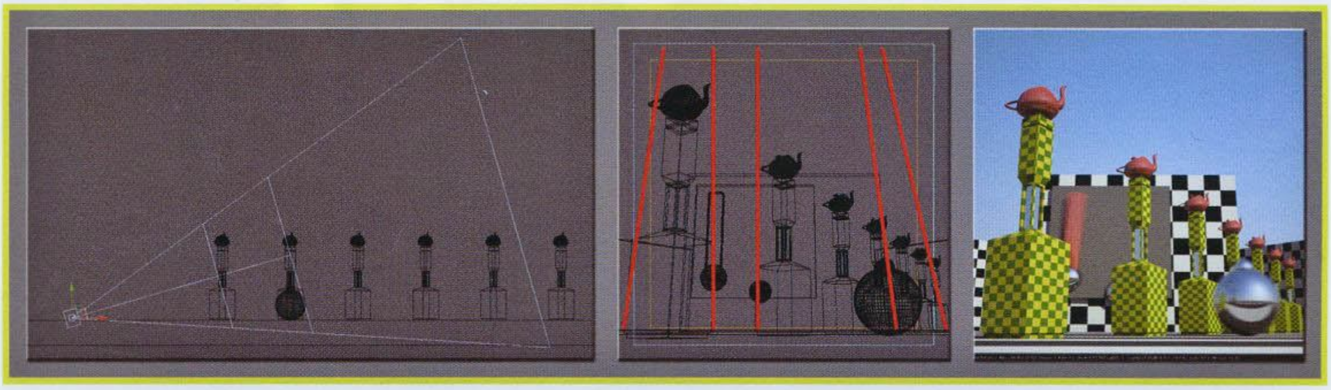


In order to solve the problem of falling lines, the only method is to de-center the objective and hence avoid the rotation of the camera and maintain the parallelism between the focal plane and the plane of the subject.



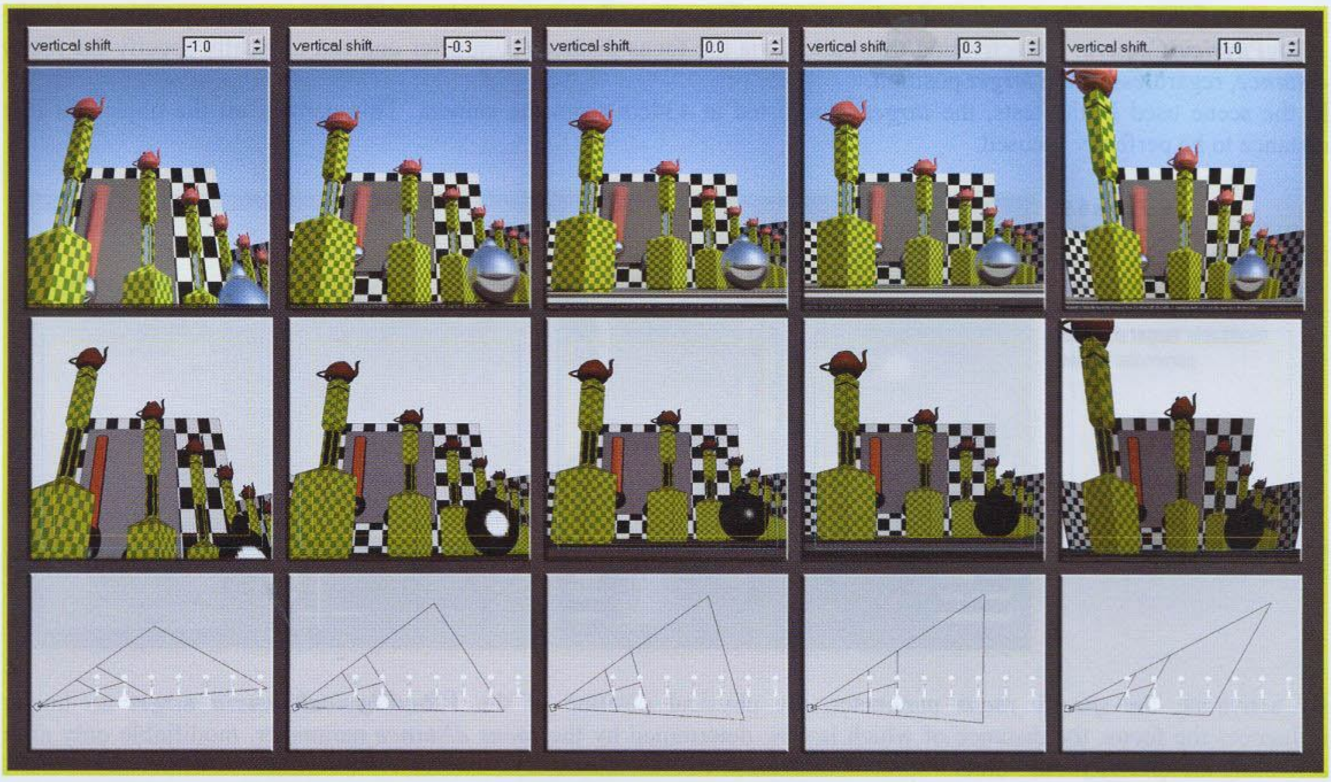
■ **Figure 6.87**
A Shift & tilt objective used for correcting the problem of falling lines. In geometry these two ways of representing 3D elements are called tilted perspective and vertical perspective.

The scene used is the same as the previous ones, with the modification of the camera position alone.



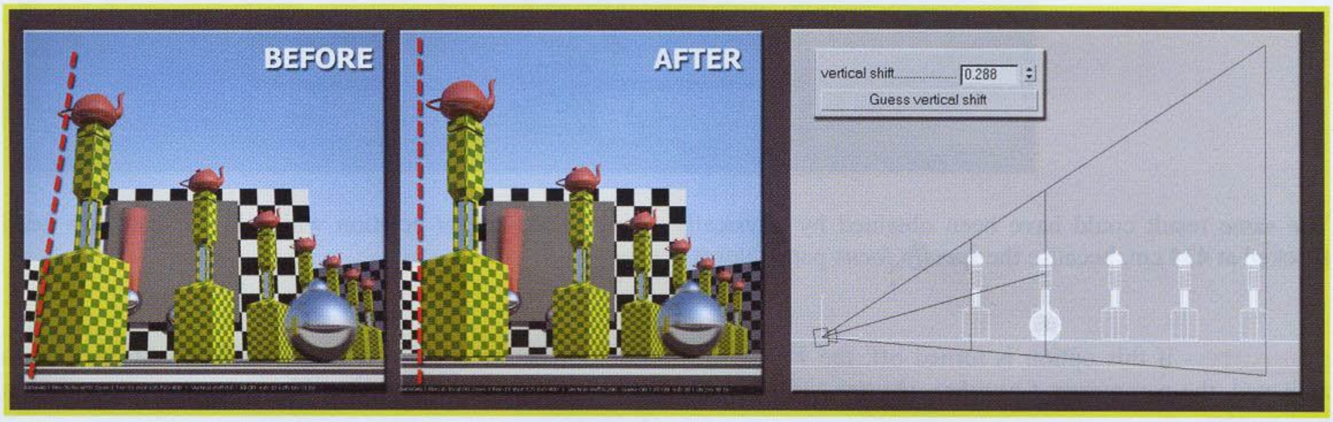
■ **Figure 6.88**
The ***target*** is higher than the camera itself. The falling lines phenomenon reappears.

In order to represent the problem of falling lines, the ***vertical shift*** parameter is used. Positive values rotate the virtual film plane in a clockwise direction. Vice versa, negative values rotate it in an anticlockwise direction.



■ **Figure 6.89**
Negative and positive variations of the ***vertical shift*** parameter.

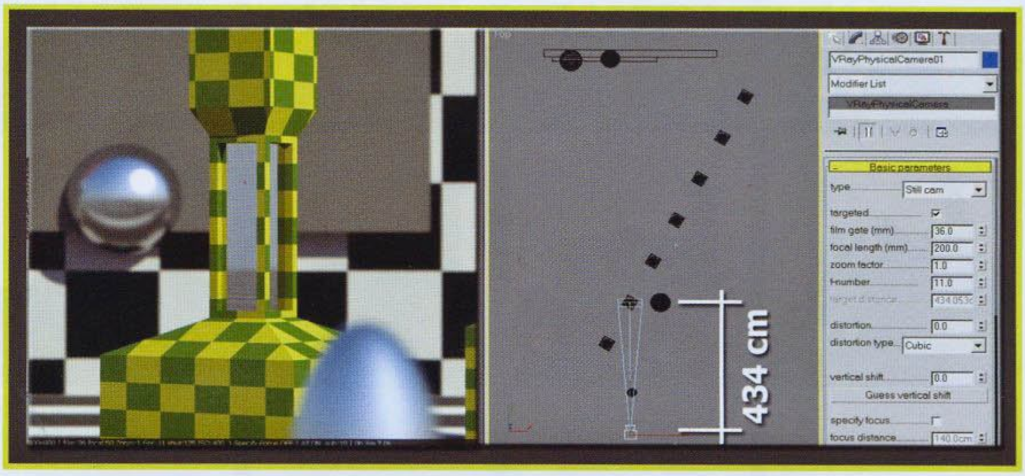
This time, in relation to ***distortion***, the viewport shows exactly what is to be rendered, with the correction fulfilled by the ***vertical shift*** parameter applied to 3D models.
Moving on, there is a button called ***Guess vertical shift*** located within the ***VRayPhysicalCamera***. Thanks to this parameter, *VRay* automatically finds the right ***vertical shift*** value. In the previous example, the value is 0.288.



■ **Figure 6.90**
The ***Guess vertical shift*** parameter automatically corrects the problem of falling lines.

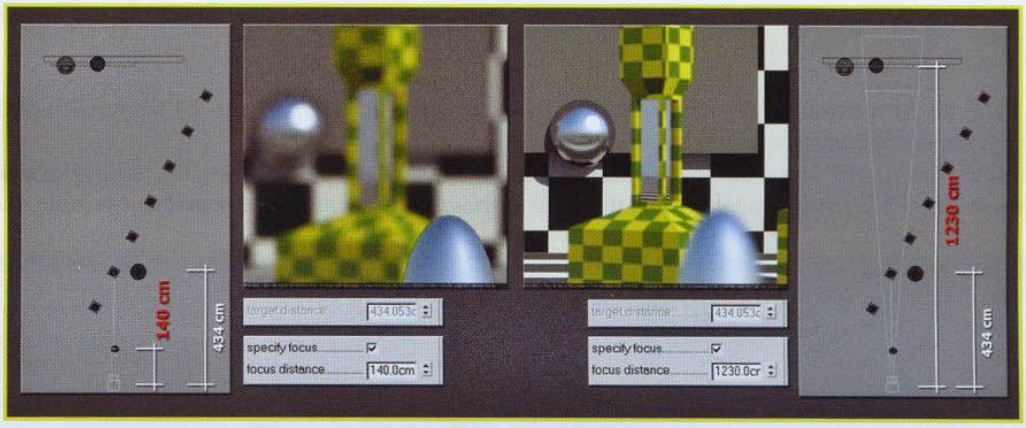
The correction applied by the ***vertical shift*** parameter can be seen.

specify focus - if deactivated, the point of perfect focus coincides with the ***target*** position of the ***VRayPhysicalCamera***. By activating this option, via the focus distance parameter one can manually specify the ***focus distance***, regardless of the ***target*** position.
In the scene used in the tests, the ***target*** is adjusted at 434cm from the camera. This permits all the objects at that distance to be perfectly focused.



■ **Figure 6.91**
The ***target*** of the ***VRayPhysicalCamera*** is manually adjusted to 434cm from the point of observation, as the non-modifiable ***target distance*** parameter shows.

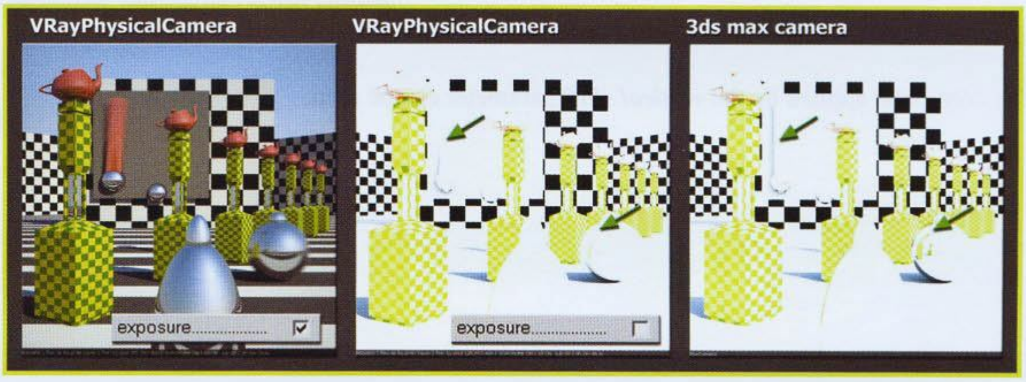
By activating the ***specify focus*** parameter, the physical position of the ***VRayPhysicalCamera target*** no longer influences the focus, the distance of which is now determined by the ***focus distance*** parameter, modifiable only after activating the ***specify focus*** value.



■ **Figure 6.92**
The modification of the focus point is now managed by the ***focus distance*** parameter.

The same result could have been obtained by physically changing the ***target*** position which in our case has been blocked at 434 cm, because the ***specify focus*** parameter is active.

exposure - if this option is turned off, the ***VRayPhysicalCamera*** behaves like a normal 3ds Max camera. If it is activated, the parameters ***f-number***, ***shutter*** and ***ISO*** affect the render exposure and the ***VRay*** camera behaves like a real camera.

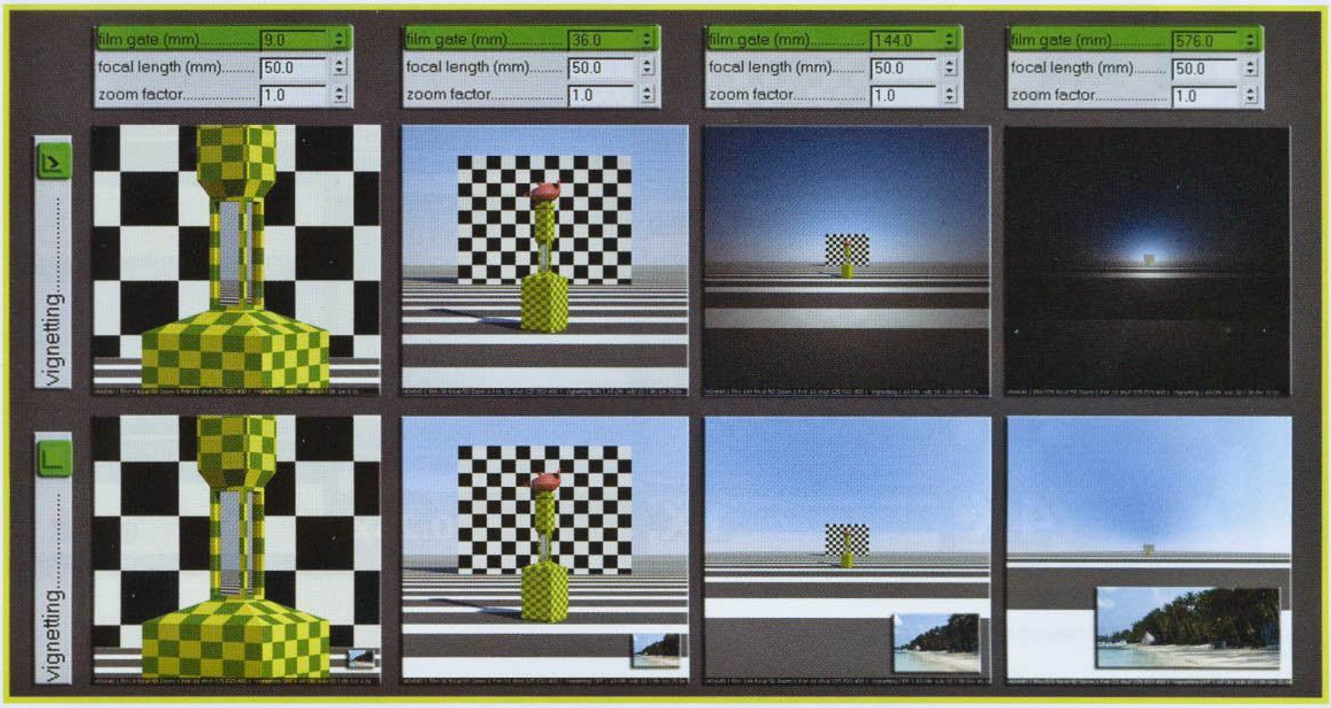


■ **Figure 6.93**
The effect due to the deactivation of the ***VRayPhysicalCamera***.

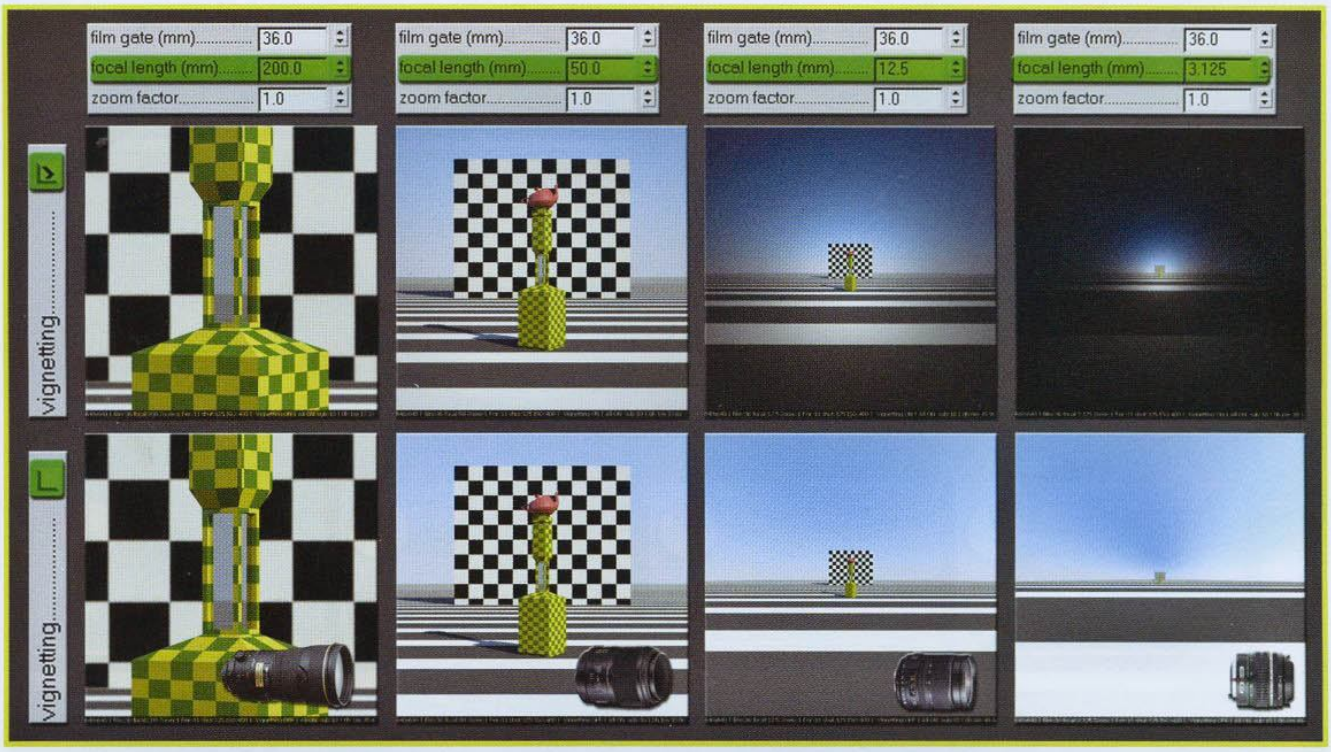
The studio scene is illuminated by a ***VRaySun***, a light which simulates the real behavior of the Sun.

As in nature, also within *VRay* the Sun has a high luminosity intensity which is blinding if not observed through a physically correct device which manages exposure. This situation is obtained both by deactivating the ***exposure*** function of the ***VRayPhysicalCamera*** and by using the 3ds Max camera. In this last case one can see the lack of ***DOF*** and ***motion blur***, even though the level of overexposure is very high.

vignetting - by activating this option, the ***vignetting*** effect is applied to the final render, a common phenomenon in cameras.
First of all we will analyze the ***vignetting*** effect when ***film gate*** and ***focal length*** parameters are changed.



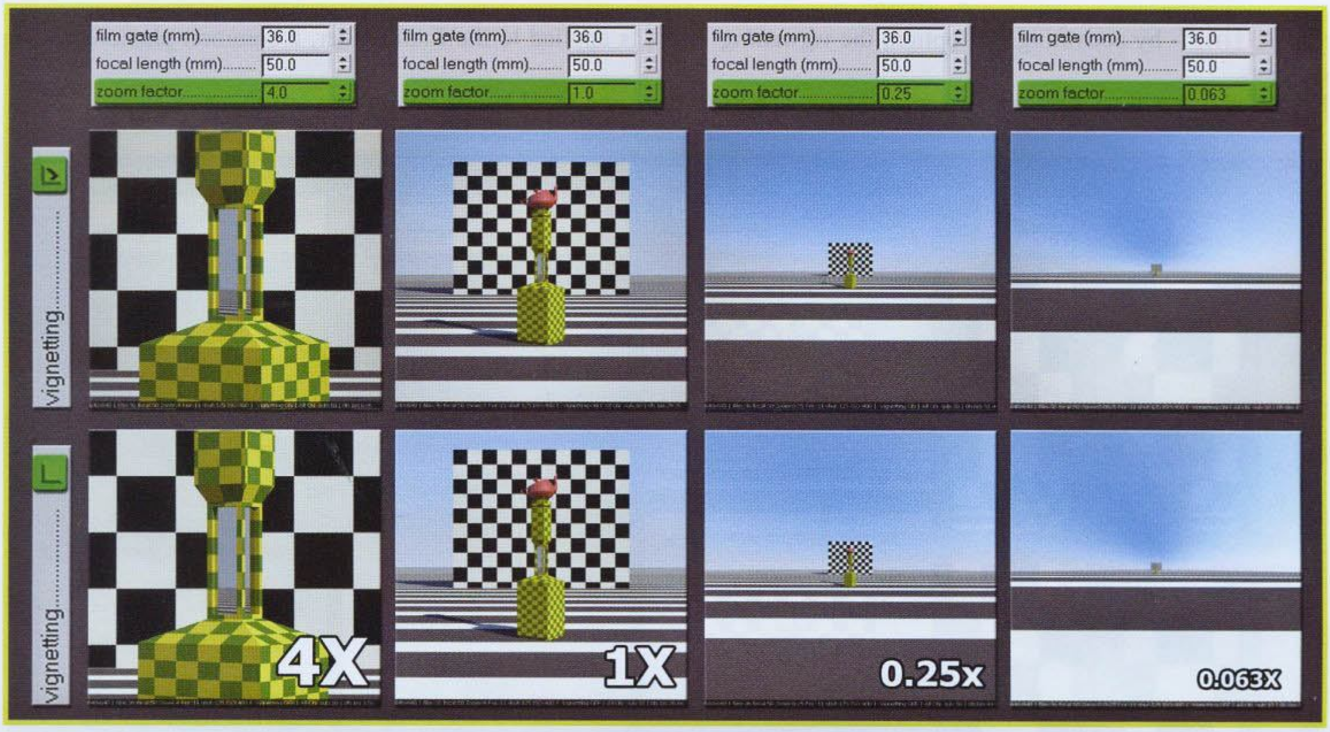
■ **Figure 6.94**
Change in vignetting due to the increase of the ***film gate*** setting.



■ **Figure 6.95**
Change in vignetting due to the lowering of the ***focal length*** setting.

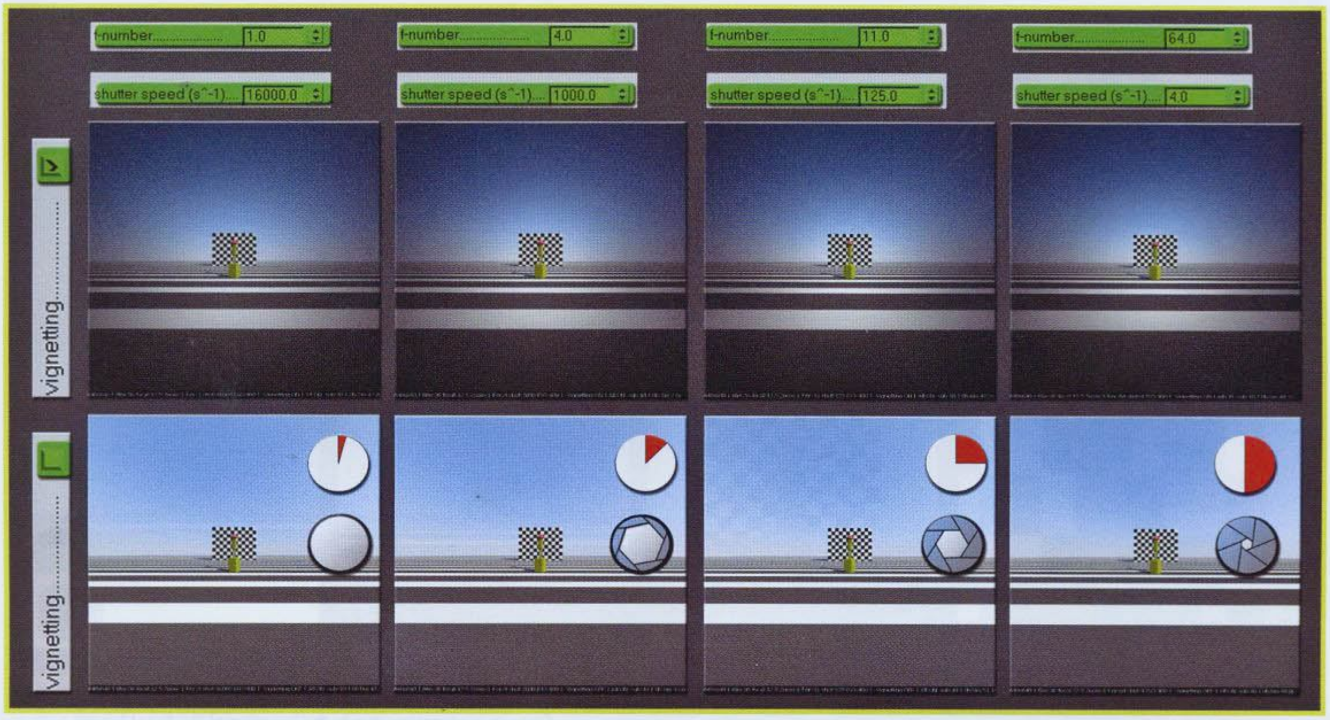
Vignetting increases both when ***film gate*** is increased and when ***focal length*** is diminished, hence using wide angle objectives.

A different situation ensues when the ***zoom*** parameter is changed.



■ **Figure 6.96**
Change in vignetting due to lowering the ***zoom factor*** parameter.

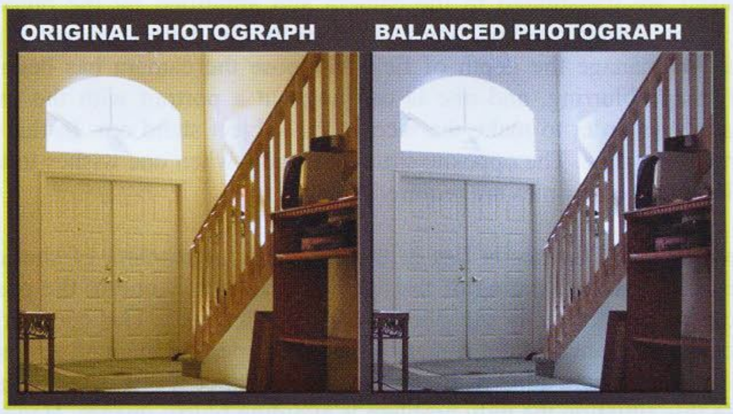
As far as the activation or deactivation of the ***vignetting*** option is concerned, the ***zoom factor*** does not influence the final image.
Lastly, the ***f-number*** and ***shutter speed*** parameters. These two parameters are closely related and by changing one of the two, in order to have the same amount of exposure, the other one must also be changed. Here some tests are shown where we have decided to maintain the same amount of exposure changing aperture amount and shutter time.



■ **Figure 6.97**
Change in vignetting due to the change in aperture and shutter time.

Changing the ***f-number*** and ***shutter speed*** settings does not influence the ***vignetting*** effect in any way.

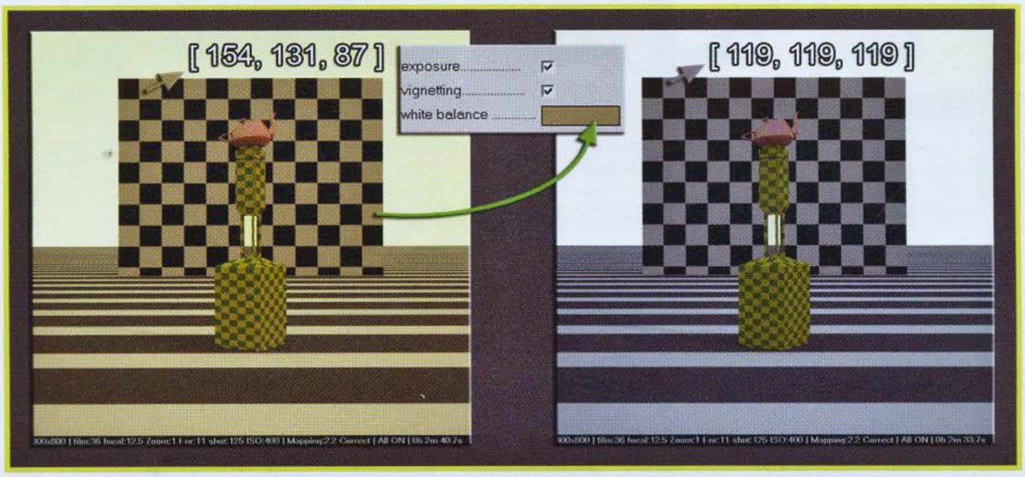
white balance - imagine observing a white sheet of paper in broad daylight. If we measure its color with a device, this gives yellow/light blue hue, as it is affected by the Sun's light and by the sky. By observing it with ones eyes, the sheet, strangely enough, appears white. If we look at the same sheet of paper in a room illuminated with an incandescent lamp, it will still appear white, although in truth it contains a yellowish dominance. What actually happens is that the eye compensates the color white, in other words it makes us see as white what the human mind knows should appear so, even though, technically speaking, it is not. One can simulate this effect also within a camera.



■ **Figure 6.98**
Correction of white, carried out automatically with a digital camera. This permits results without dominant colors and gives a neutral balance.

In traditional analogue photography, to correct color temperature, there are two tools available: films calibrated for specific situations and filters for correction.
Digital cameras, instead, were born to give everyone the possibility of taking good quality photos and provide an automatic light correction system which preserves natural colors as much as possible. For example, some digital cameras have specific presets for daylight, cloudy daylight, neon light (fluorescent), flash light or incandescent lamp-lit situations.

The ***white balance*** parameter therefore allows one to change the general color of the render, just like a digital camera would. The objects which have the color placed within this slot are considered white. One can, for example, compensate the yellow color produced by the ***VRaySun***.



■ **Figure 6.99**
An orange sky obtained by increasing the ***turbidity*** parameter in ***VRaySun***. In order to correct the color, the ***white balance*** parameter has been on the "white" color of the checker pattern.

In the previous image, the user knows in advance that the color of the chessboard is black and white, as he knows the materials used to create it in the Material Editor. As the sky has a high yellow component, the "white" of the wall is no longer white but yellowish. If one uses the color taken from the render as the parameter for ***white balance***, in the render "white" will finally become real white and as a result all the other colors do not suffer from the dominant yellow tone and appear neutral.

shutter speed - this represents the opening speed of the shutter. It is expressed in seconds^-1. For example, by setting a value of 250, this is interpreted by *VRay* as 1/250th of a second.
The ***shutter speed*** parameter is closely related to the aperture. It is therefore necessary to adjust one of the two parameters and, in order to find a correct amount of exposure, modify the second one. Which of the two, ***f-number*** and ***shutter speed***, should be considered as the main parameter?

At this point two concepts must be introduced: "aperture priority" and "shutter priority".

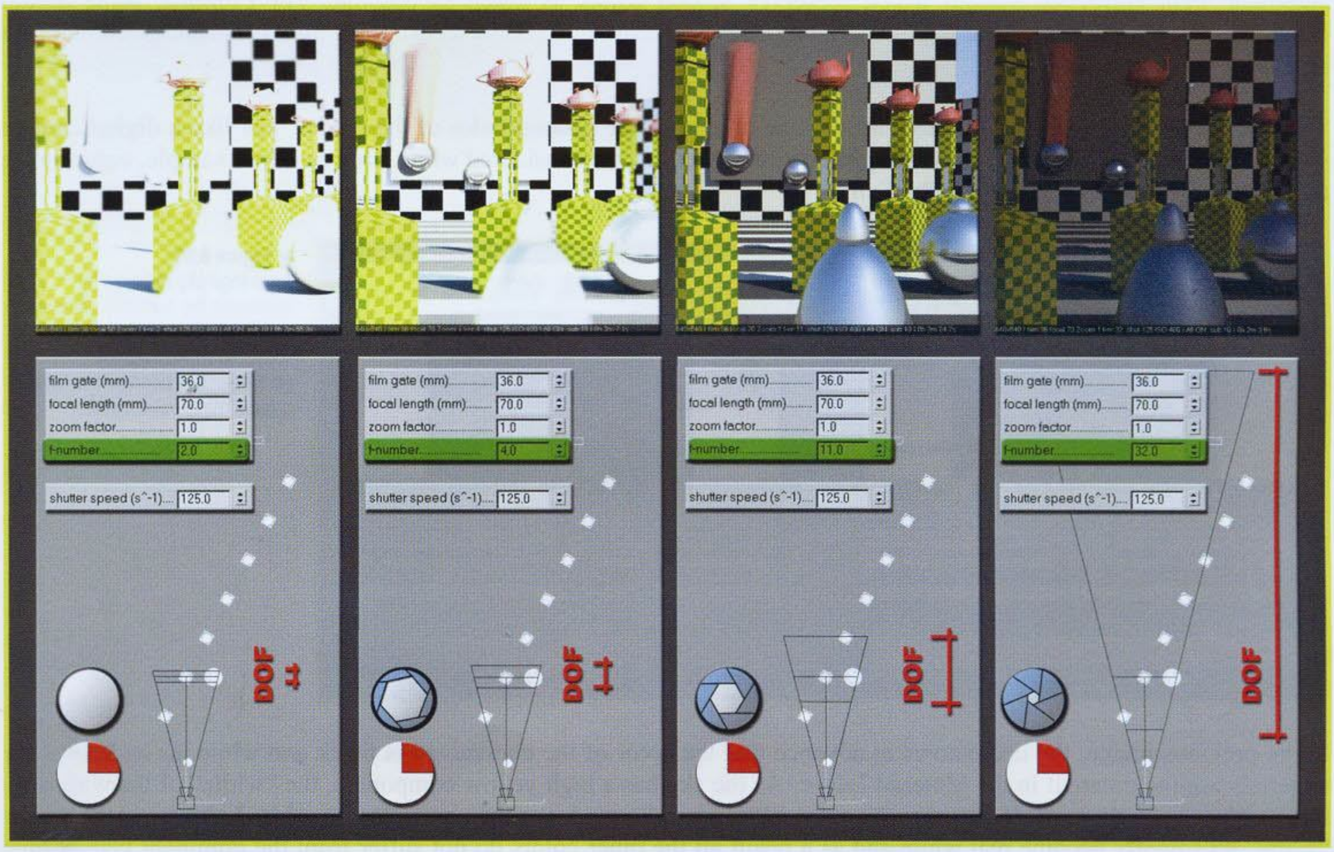
Aperture priority: used for controlling depth of field.

Shutter priority: This mode is used mostly for avoiding blurring when the subject is moving.

As represented in the well-known EV chart, there can be a great number of aperture/shutter pairs apt for the same degree of exposure, but all of them produce very different effects as far as ***DOF*** and ***motion blur*** are concerned.
Imagine a very common situation, handheld photography. If exposure time becomes too long, one risks blurred images, as the vibrations of ones hand are transmitted to the camera and amplified by the objective. So when using tele objectives, one should use short shutter opening times. As well as the danger of blurry images, there may be the decision of the photographer to manage the depth of field. Suppose the camera has been setup on a tripod, hence eliminating the problem of image blurring, and one is carrying out a portrait with the precise aim of blurring the background as much as possible in order to make the face of the subject stand out. In this case we will need a good aperture opening and a short shutter time. The opposite situation occurs when one wishes to take photographs of scenes with low luminosity where long exposure times are required.

In modern reflex or compact cameras, the secret lies in the automatic adjustment of shutter time and aperture. This means that the camera, after an automatic reading of the light meter, chooses one of the possible shutter/aperture pair settings starting from a fixed point chosen by the photographer, at his will. The operator chooses the shutter time because he needs that particular timespan and the camera adjusts the aperture amount, or instead an aperture setting is chosen because it is needed and the camera adjusts shutter time: for instance if one is taking photos of an outdoor scene, while children are playing. In this situation the only problem is avoiding blurry images. Hence one sets a shutter time of 1/250th of a second and the camera takes care of the rest.

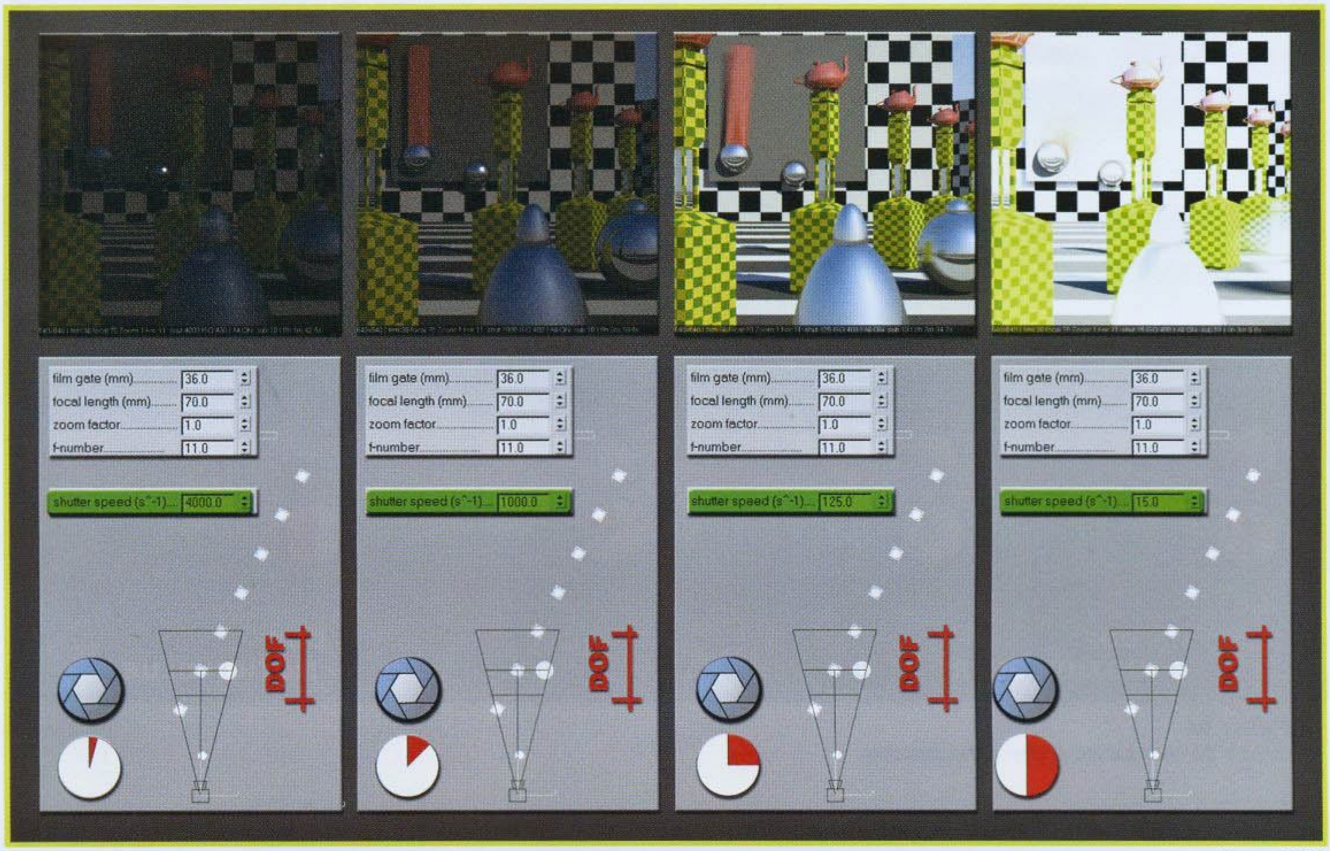
In the following tests, we will see separately how both the aperture or shutter time affect an image. We will then go on to analyze their symbiotic behavior.



■ **Figure 6.100**
Variation of the ***f-number*** only. Small apertures produce a high depth of field.

As one might expect, by changing the ***f-number*** parameter alone, if ***shutter speed*** is not changed as well, one obtains a reduced or excessive film exposure. One can observe how the depth of field changes with aperture variation (within the limits of possibility, because of the high luminosity), both in the render and the viewport. ***Motion blur*** instead remains unaltered.

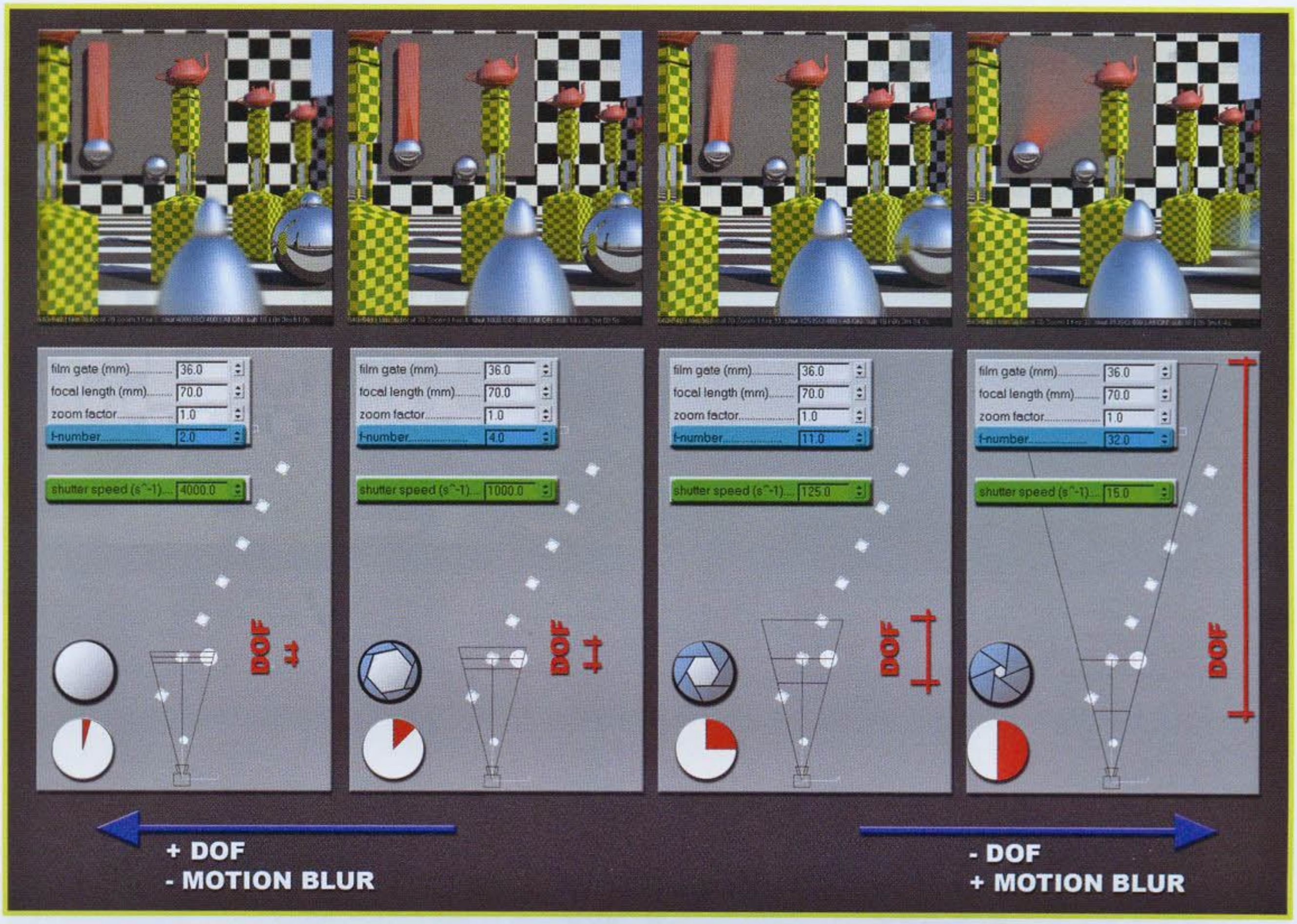
By changing ***shutter speed*** alone, also in this case there are important variations.



■ **Figure 6.101**
Variation of the ***shutter speed*** parameter alone.

By decreasing exposure time, that is, using higher ***shutter speed*** settings, the image is obviously darker because the film is exposed to light for a shorter time. The result is similar to what would happen if one used small aperture settings. The main difference lies in the ***DOF*** and ***motion blur***. The planes which represent the borders of the depth of field, which can be seen in the viewport, do not change with ***shutter speed***. This means that exposure time has nothing to do with the ***DOF***. Instead one can see the effect of the shutter on ***motion blur***. In this case, by changing exposure time, one can "freeze" movement and avoid the typical stripes of ***motion blur***.

It appears to be clear that, in order to maintain the same exposure, if one changes ***shutter speed***, one must compensate the different amount of exposure with a new ***f-number***. In this case one obtains renders with the same amount of exposure, but with a ***DOF*** and ***motion blur*** which are very different, always in relation to the focal length.



■ **Figure 6.102**
Variation of the ***f-number*** and ***shutter speed*** parameter.

shutter angle - this represents the angle of the shutter in degrees for cameras. In order to calculate the exposure time of the film, one can use the following formula:

$$\textbf{Exposure in seconds} = \frac{\textbf{Shutter angle}}{\textbf{360 * fps}}$$

where:

Shutter angle: amount of opening of the shutter in degrees.
fps: speed of the film. It can be 24, 25 or 30 fps.
360: this is a constant. It represents a round angle.

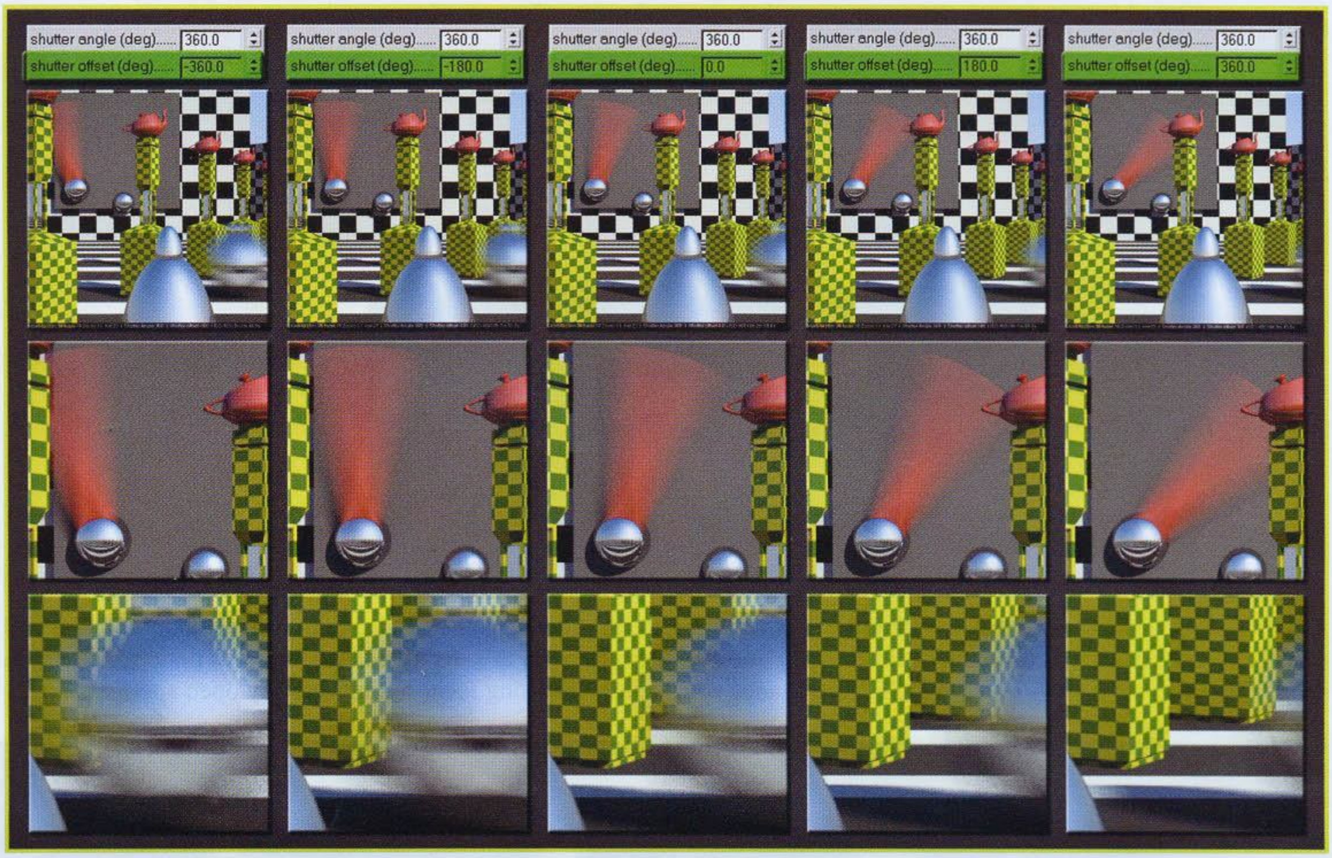
If one wanted to calculate the ***shutter angle***, knowing the amount of exposure:

Shutter angle = Exposure * 360 * fps

In the current scene, where the animation is of 30 fps, an exposure of 1/125th of a second corresponds to 86.4° ***shutter angle*** with ***f-number*** 11; an exposure of 1/4000 corresponds to 2.7° with ***f-number*** 2. At the maximum aperture possible, namely 360°, exposure time is equivalent to 1/30 sec., which is the ***frame rate*** of the film.

■ **Figure 6.103**
The ***shutter angle*** is the parameter used in movie cameras for managing shutter time. In this example the same results have been obtained by using a ***Movie camera*** and a ***Still camera***.

shutter offset - this represents the movement in angles of the shutter for movie cameras.

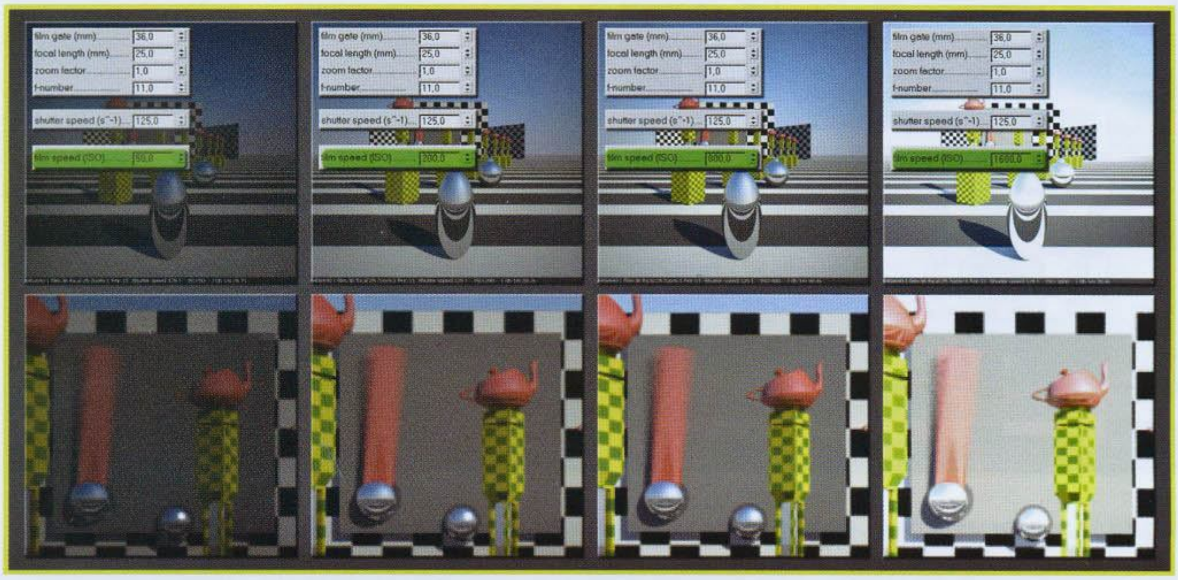


■ **Figure 6.104**
The ***shutter offset*** moves the ***motion blur*** generated by the ***VRayPhysicalCamera*** forwards or backwards.

Thanks to this parameter it is possible to change the ***motion blur*** generated by the shutter, by moving it clockwise or anticlockwise.

latency - for digital CCD cameras, this represents the latency of the electronic shutter measured in seconds.

film speed (ISO) - this indicates the sensitivity of the film. At low settings the film is not very sensitive. If this is the case, at equal ***f-number*** settings the film will have to be exposed for longer compared to the more sensitive film. On the contrary, at equal ***f-number*** and ***shutter speed*** settings, by using more sensitive films, one obtains brighter and clearer images.



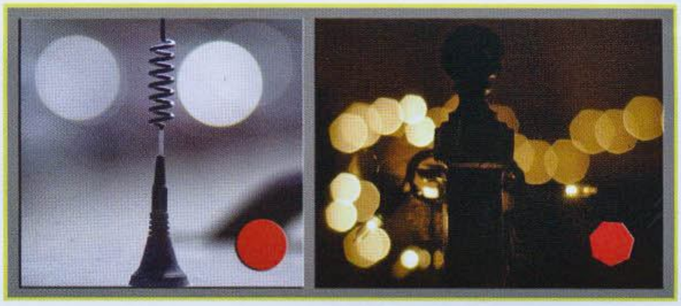
■ **Figure 6.105**
Variation in luminosity due to modifying the ***ISO*** parameter only.

The use of films with different levels of sensitivity affects exposure significantly, whereas neither the ***DOF*** or ***motion blur*** are affected.

Bokeh effects

blades - this parameter influences the ***Bokeh*** effect directly, in the so-called blurred zones, especially highlights.

■ **Figure 6.106**
The ***Bokeh*** effect fulfilled on the highlight of the sphere.



As one can see from the images, the ***Bokeh*** effect can take on different shapes and is able to produce both circular and polygonal geometries. It may seem of little use to bring ones attention to this detail, but this effect is noticeable only in situations which have a low level of DOF, for instance when using very long focal settings or very open apertures.

■ **Figure 6.107**
The ***Bokeh*** effect fulfilled on the ***highlight*** of the sphere.

In the previous image, as well as the increase in rendering time due to the greater amount of ***DOF*** in the scene, one can easily see the ***Bokeh*** of the highlight produced by the chrome sphere.

If the small box next to the ***Bokeh*** parameter is deactivated, the result is a perfect circle. This means that the segments which make up the aperture are not taken into account.

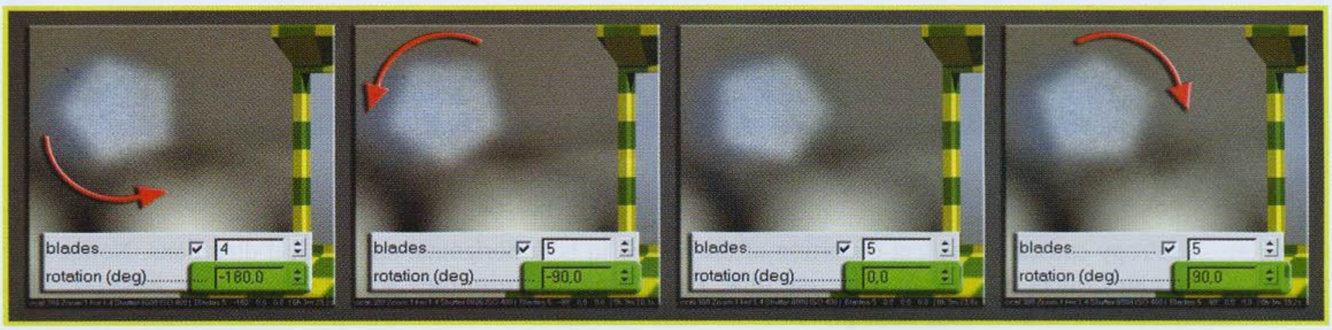
■ **Figure 6.108**
Activation and deactivation of the blades for the ***Bokeh*** effect.

In order to manage the number of ***Bokeh*** blades, one must activate the ***blades*** parameter and define the number of blades.

■ **Figure 6.109**
Bokeh effects of different shapes and sizes.

Starting from a number of 3, one can define the number of blades to be used in ***Bokeh*** as one prefers. Notice the way geometries change in the spot indicated by the green arrow.

rotation - this defines the rotation of the blades of the aperture; this means a rotation of the ***Bokeh*** effect. One can rotate it clockwise or anticlockwise.



■ **Figure 6.110**
Bokeh rotation.

center bias - this parameter allows one to choose the shape of the ***Bokeh***, meaning the thickness of its sides.

■ **Figure 6.111**
Change in shape and profile of the ***Bokeh*** effect.

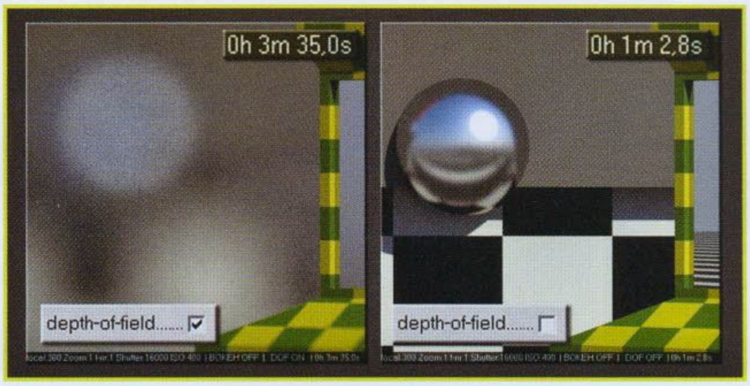
The way the ***Bokeh*** effect dies out with this parameter can be defined quite precisely. One can produce a gradient which attenuates the ***Bokeh*** from the outside moving inwards towards the center. One can also create a ring of variable thickness. In this last situation ***center bias*** has a positive value for its setting, instead for creating the aforementioned gradient, it needs negative values.

anisotropy - this permits the ***Bokeh*** effect to be deformed, both horizontally and vertically. It is useful for simulating anamorphic lenses, used in some objectives.

■ **Figure 6.112**
Bokeh deformation.

Changing anisotropy causes an evident distortion of the ***Bokeh*** in both horizontal and vertical directions. Notice how this also causes an evident increase in rendering time. By moving the ***anisotropy*** setting closer to the limit [-1, 1], a peculiar underexposure of the render takes place.

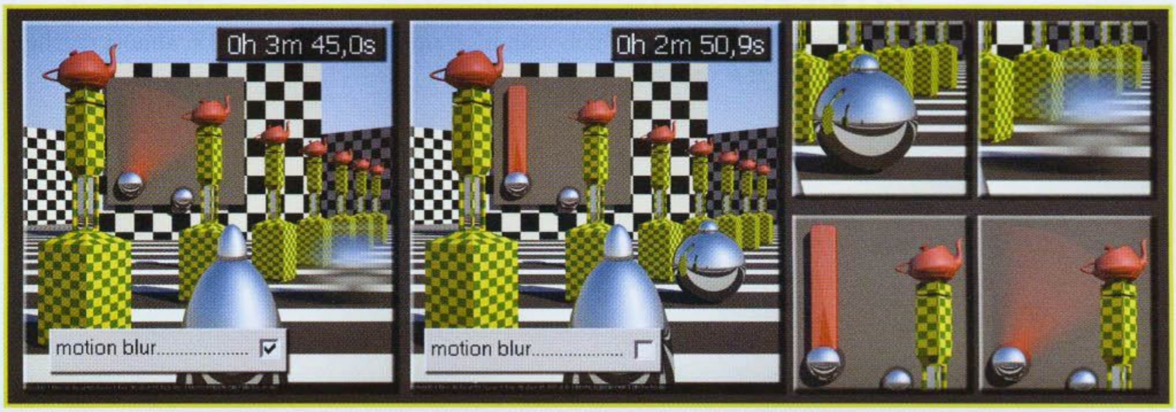
depth of field - this parameter activates or deactivates the ***DOF***.



■ **Figure 6.113**
Activation and deactivation of the *DOF*.

Activating the ***DOF***, with subdivs set to 10, leads to an increase in rendering time of 3.5 times!

motion blur - this parameter activates or deactivates ***motion blur***.



■ **Figure 6.114**
Activation and deactivation of *motion blur*.

Activating ***motion blur*** has increased rendering time, although not as much as ***DOF*** activation. This is mainly because, unlike the ***DOF*** which is applied to the almost whole image, ***motion blur*** is only applied to very few geometries in the scene. Notice how ***motion blur*** applied to the red parallelepiped creates an perfect circle arc. The quality of this circle and its segmentation is defined by the ***Geometry samples*** parameter in the ***VRay:Camera*** rollout, in the Renderer window.

■ **Figure 6.115**
Definition of the number of *motion blur* segments.

subdivs - this parameter determines the number of samples used for calculating the ***DOF*** and ***motion blur***, and therefore defining their level of noise.

■ **Figure 6.116**
Renders with different ***subdivs*** values.

There are two main observations to be made after observing the previous test. The first concerns the difference of ***DOF*** and ***Bokeh*** quality. With very low ***subdivs*** settings, like 2 or 6, the ***DOF*** is grainy and dirty. With high values, like 16, the ***DOF*** is smoother and the ***Bokeh*** has gained brightness and precision, as we can see along the edges of the pentagon. The second observation concerns rendering time, which with high ***subdivs*** values, like 18, almost reaches 10 minutes, as opposed to 1 min. with a low setting of 6 ***subdivs***.

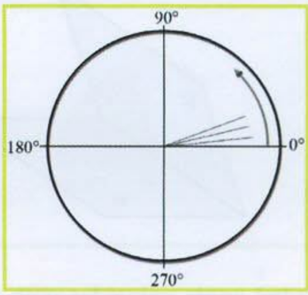
VRay Light and VRay Shadows

ILLUMINATION ENGINEERING THEORY NOTIONS

The ability to see depends on the amount of light in the environment one is in. The arrival of artificial forms of light has made it necessary to quantify the levels of light useful to visual work and establish units of measure for photometry. The fundamental quantity is light intensity and the unit for measuring it is the candela. However, before introducing the concept of candle properly, it is necessary to understand other parameters, both of geometric and physical character, such as the radian, dihedral angle, angoloid, solid angle, steradian, the lumen and so on.

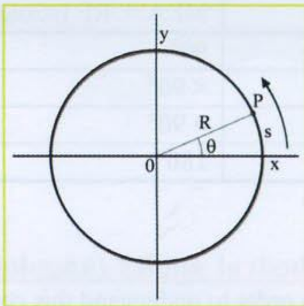
Geometrical parameters

Radians - in geometry angles can be measured in a few different ways. The most common one is undoubtedly the sexagesimal degree, usually indicated by the symbol °; 1° represents 1/360th of the circumference of a circle.



■ **Figure 6.117**
Sexagesimal degrees, the most common way of measuring angles.

Another method, less common in everyday life but preferred by mathematicians and physicists, is the radian. Here it is explained: We shall consider the angle θ in the following image, and we shall call the portion of the circumference it refers to s:

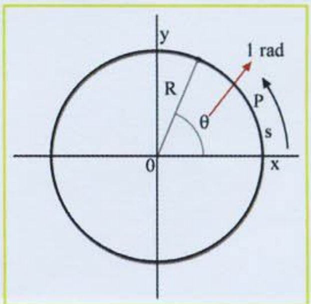


■ **Figure 6.118**
A 360° angle corresponds to 2n rad.

The measurement in radians of the θ angle consists of the ratio between the length of the arc s and the measure of the radius R. So:

$\theta = s/R$

Evidently, an angle of 1 radian, which is also written as 1 rad, is an angle which subtends an arc as long as its radius:



■ **Figure 6.119**
By definition 1 radian is an angle, on a circumference centered on its vertex, which subtends an arc of the same length as the radius.

How much is a round angle equivalent to in radians? As the arc which subtends a perigon is the whole circumference which measures 2πR, the measure of a round angle is therefore:

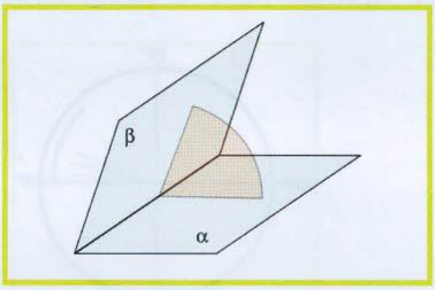
$2\pi R/R = 2\pi$

Notice how the radian, being a ratio between two lengths, is a measure without physical dimensions. It is in fact a pure number.

From here we can find a correspondence with the main sexagesimal angles:

SEXAGESIMAL DEGREES	RADIANS
0°	0
90°	1/2 π
180°	π
270°	3/2 π
360°	2 π

Dihedral angle - a dihedral angle is an extension of the concept of angle in a three-dimensional environment. It is defined as a portion of space located between two semi-planes called faces, which share a line of intersection.

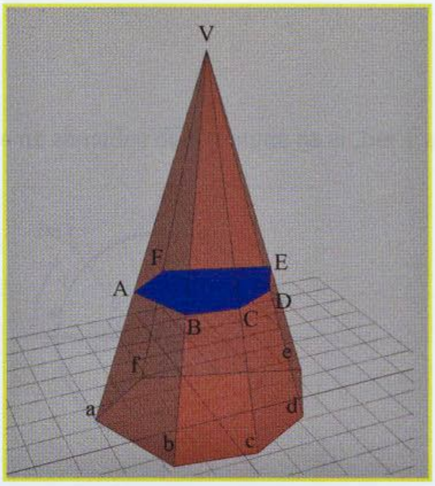


■ **Figure 6.120**
Graphic representation of a dihedral angle between two semi-planes.

The measurement of a dihedral angle coincides with the measurement of a plane angle which has been obtained by making a section of the dihedral angle with a plane perpendicular to the line of intersection. On the basis of its measurement, one can classify dihedral angles in:

NAME	MEASURE (sexagesimal degrees)	MEASURE (radians)
Right angle	90°	π/2
Acute angle	< 90°	< π/2
Obtuse angle	> 90°	> π/2
Straight angle	180°	π

Solid angle with polyhedral angles (angoloid) - this is the area of space between the dihedrals, its corners having a common vertex. In order to understand this concept better, observe the images.

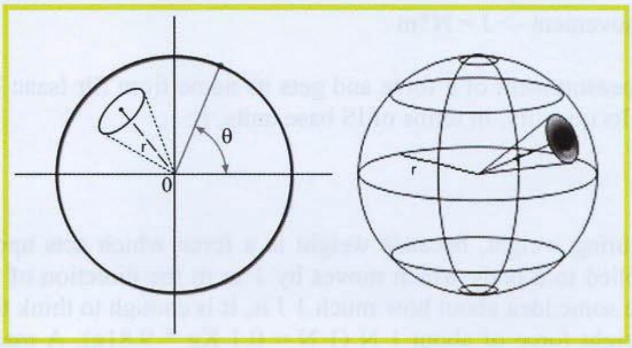


■ **Figure 6.121**
This concept we shall call "angoloid", is the figure formed by all the half-lines with the origin of *V* which pass through different points of a given polygon. The whole of these faces is called contour or surface of the angoloid, whereas the intersection of two of these spaces, having a common edge, is called dihedral of the angoloid.

Supposing we have a convex polygon ABCDEF and a point V, anywhere in space, draw the half-lines which pass through the vertexes of the polygon. The convex angoloid is the unlimited part of space delimited by the angles AVB, BVC, AVD, CVD, DVE, EVF, FVA and which contain the ABCDEF polygon. The common origin of the half-lines Va,Vb,Vc,Vd,Ve and Vf is the vertex of the angoloid, the half-lines Va, Vb, Vc, Vd, Ve and Vf are its edges, and the angles formed by the vertex V and the edges are its faces.

Solid angle - the concept of solid angle re-uses the definition of angoloid, although technically they are not the same thing. Also the concepts concerning radians which we have seen before are used again too.

Imagine a plane angle, the classic angle of $\theta = 60°$, like the one shown in the image.



■ **Figure 6.122**
Scheme which illustrates the concept of plane angle and solid angle.

Place its vertex in the center of a circle. In this case its two segments will cut the circumference and intersect it. The total length of a circumference is equal to the diameter multiplied by π.

$c = 2\pi * r$

so roughly, 6.28* π. Obviously there is a certain proportion between the amplitude of the θ angle and the portion of the circumference which has been cut. An angle which cuts the circumference of a circle and is as long as the radius if the circle itself, can be used as a unit of measurement and goes by the name of **radian**.
Image that the circle in the image corresponds to the vertical cross section of a sphere. Now draw a circle on a spherical surface, hence delimiting a spherical cap. From the center of the sphere, draw a cone, with its vertex in the center of the sphere itself and as a base the spherical cap which has just been drawn.
In the same way that a plane angle occupies part of the plane which surrounds the center of the circle, with the same criterion we can say that the cone we have just described occupies a part of the space surrounding the center of the sphere. The solid which has just been created goes by the name of **solid angle**. If the base of the cone is not a spherical cap, but a polygon, the cone becomes a pyramid and the solid angle is called **angoloid**. If the surface of the spherical cap which acts as a base for the cone corresponds to an area of the value of r^2, one can assume this solid angle to be the unit of measurement of solid angles. This unit is called **steradian** (symbol: str or sr). In Greek "stereron" means solid, or in three dimensions. One can therefore define the steradian as the solid angle, at the center of a sphere with a radius r, subtended by a portion of the surface of the sphere which has an area of r^2.
The solid angle Ω is mathematically defined as:

$\Omega = A/R^2$

where A is an area on a spherical surface with a radius of R.

This definition is similar to that of the θ angle in radians.

$\theta = s/r$

where s is a length of an arc of a circle with a radius of r.

If one takes a perfect sphere as a reference, its surface measures $4\pi r^2$. The solid angle which takes up all the space must therefore be equal to 4π str. In order to have a measurement in square degrees, one multiplies the value in steradians by $(180/\pi)^2$, which is equal to 3282.8. The whole sphere corresponds to about 41,253 square degrees.

Physical parameters

So far we have analyzed the geometrical parameters which will be needed further on in the book for understanding values used in illumination engineering, such as the candela, luminous intensity, the lumen, luminance or radiance, radiant flux, and so on. Before facing these subjects it is best to clarify a few physics terms.

Mechanical work - mechanical work is defined as the product of a force applied to a body and the displacement which it undergoes because of this force.

Mechanical work = force*movement -> J = N*m

The Newton is the unit of measurement of a force and gets its name from Sir Isaac Newton, as a reorganization for his work on classic mechanics. Its quantity, in terms of IS base units, is:

$N = 1Kg*m*s^{-2}$

It is also the unit for measuring weight, because weight is a force which acts upon between two bodies because of gravity. A force of 1 N applied to a body which moves by 1 m in the direction of the force, carries out a mechanical work of 1 J. In order to have some idea about how much 1 J is, it is enough to think that a body of 1 Kg of mass, subject to Earth's gravity, has a weight force of about 1 N (1 N = 0.1 Kg * 9.81g). A mass of 1 Kg, near the surface of the Earth, weighs 9.81 N. One can therefore deduce that by lifting the same object by 1 m, one fulfils 9.81 J worth of work. In a very empirical and intuitable fashion, one could say that where we find work, there must be some energy involved, and the mechanical work can be used to measure the energy itself.

Energy - as we have just described, when one exerts a force in order to make something move, deform it, block it or get it running, the result depends not only on the intensity of the force but also on the zone something must move across. The product of force by displacement is called mechanical work. Work allows one to measure energy which is given in exchange when one exerts a certain force in order to obtain a desired result: the result does not depend only on the force or the displacement but on the product of the two, that is, the energy which is transferred. An example of transferred energy is that which is transferred from a finger to the elastic band in a sling: it does not only depend on the force applied by the finger on the elastic band, but also on the stretching of the elastic band itself, its displacement in fact.
Energy takes on many different forms, and can transform from one type to another. For instance, the muscular energy of the finger becomes the elastic energy of the sling when one pulls the elastic band back to stretch it. The importance of energy is therefore understandable as it allows one to carry out work, in other words to apply force in exchange of a useful result. In nature there are many other types of energy as well as mechanical energy. It may sometimes seem as if mechanical energy disappears or is created from nothing: in actual fact it is transformed into other forms of energy.

- When a car brakes, its kinetic energy is transformed into thermal energy: the wheels, the road and the brakes themselves become hotter.

- In a burning candle, the chemical energy turns into thermal energy and into radiant energy from the flame.

- The rays of the Sun and of a lamp emit heat: this is radiant energy transforming into thermal energy.

Energy is measured in the moment it transforms or is passed from one object to another and there are two ways to measure it: by calculating the mechanical work carried out or by measuring the heat transferred.

Mechanical work is carried out when a force is applied for a while or produces displacement: work is the product of force by displacement. For example, to lift an object up to a certain height, one must apply a force upwards, equal to the weight-force of the object.

Heat instead, is the energy transferred to an object which make its temperature increase and is calculated by multiplying the variation of the temperature by the mass and by the specific heat capacity, which is the characteristic coefficient of the substance which the object is made of. In the IS, a unit which is often used for measuring, transferred heat is measured in calories (the symbol is cal.). With 1 calorie one can heat 1 g of distilled water at sea level by 1 centigrade; 1 cal is roughly equal to 4 J.

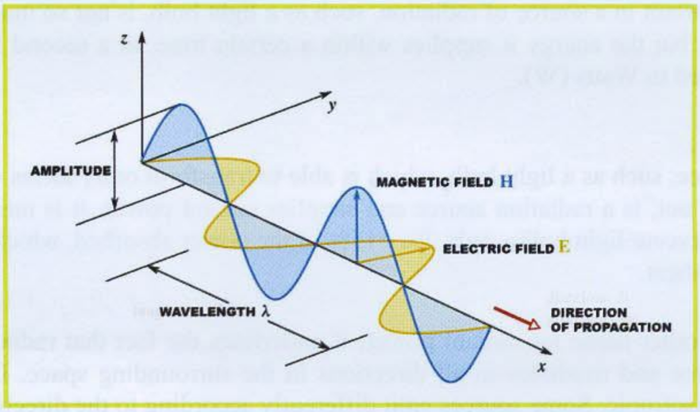
Power - often it is not only important to know how much energy is produced but also the duration of energy supply: the energy supply per unit of time is called power and its unit is the Watt, which is equal to the energy of 1 J supplied in 1 sec.

Power = energy/time => W = J / sec

Radiation - in physics, the term radiation is generally used to indicate a series of phenomena or agents which allow the transportation of energy through space. More precisely one defines radiation as any emission and propagation of energy in the shape of waves or elementary particles. Radiation can be made of elastic waves (acoustic radiation), gravitational waves (gravitational radiation) and electromagnetic waves (electromagnetic radiation), classified according to their length. Typical examples of radiation are light and heat which can be perceived by human beings via their senses.
The peculiar characteristic of all these phenomena is the transfer of energy from one point to another in space without the movement of large bodies and without the help of a medium made of matter. When the propagation of energy takes place in such a way, one defines it as radiation. The sound emitted by a pair of loudspeakers is not a radiation, but a sound wave which requires air to propagate.

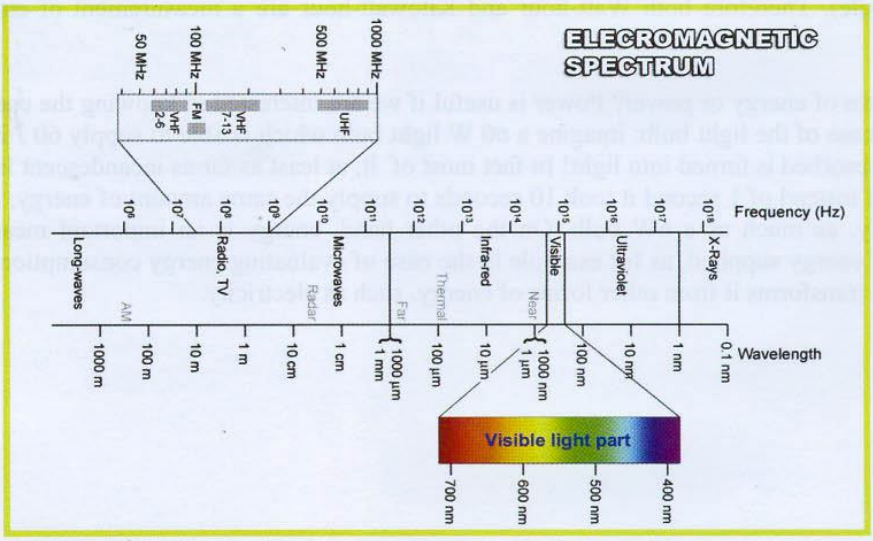
Without delving too far into the purely physical aspects of this subject, we can say that electromagnetic radiation means a form of energy which propagates through space (its speed is c = 300,000 km/sec) via the oscillation of magnetic and electric fields and characterized by three variables:

Amplitude, A
Wavelength, λ
Frequency, υ



■ **Figure 6.123**
Two important characteristics of electromagnetic radiation are: straight trajectory and the ability to pass through objects. The quantity of material which the radiation can pass through depends on the composition of the material and on the energy of the radiation. Low-energy radiations, such as visible light, do not penetrate deeply, but X-Rays and gamma rays can pass through remarkable thicknesses.

All electromagnetic waves have the same speed in empty space: 300,000 km/sec, the speed of light. The speed of light is given by the product of frequency by wavelength. Therefore these two quantities are inversely proportional: lower frequencies (1 million Hertz) correspond to longer wavelengths (300 meters), whereas higher frequencies (100 billion Hertz) correspond to shorter wavelengths (3mm). The ensemble of all electromagnetic radiations, classified according to their frequency, makes up the electromagnetic spectrum.



■ **Figure 6.124**
It is interesting to discover that only a small portion of the spectrum contains radiation visible to the human eye. Although all the waves of different zones have the same properties, the term light is used only for the part which can be seen.
A field of gamma rays, for example, is the result of radioactive disintegration. Radio waves can be generated by discharges which produce electromagnetic waves. Although one can distinguish various areas within the spectrum, one cannot define exact boundaries between them.

Radiometry

It is common experience that the light radiation produced by a lamp transports energy. The study of the transportation of this energy can be carried out from two points of view: from the point of view of radiometry, and more closely related to human vision, from the point of view of photometry. Radiometry considers radiation as a form of energy, whereas photometry evaluates its use in terms of human vision. One may say that a light radiation is radiometrically very intense, although its visual perception remains weak or even absent. This is possible if whatever is emitting the electromagnetic radiations produces them in spectral zones where the eye's response is low or inexistent, as in the case of UV rays or X-Rays.

The main radiometric quantities which we will analyze, some of which are to be used within *VRay*, are:

Radiant power	W
Radiant energy	J
Radiant exitance	W/m^2
Irradiance	W/m^2
Radiant intensity	W/sr
Radiance	$W/m^2/sr$

In order to describe and understand these values, we have reproposed the definitions of energy and power, which we have already seen earlier, but this time applied to the field of electromagnetism.

Power (W) - what is important in a source of radiation, such as a light bulb, is not so much the total amount of energy that the source can supply, but the energy it supplies within a certain time, in a second for example. This quantity is called Power and is measured in Watts (W).

Radiant power - a device, such as a light bulb, which is able to transform other forms of energy into radiant energy, whether they are visible or not, is a radiation source and supplies radiant power. It is measured in Watts like all other forms of power. For incandescent light bulbs, only 2% - 15% of the power absorbed, which is electricity, becomes light. The rest is transformed into heat.

Radiant flux - it is another name for radiant power; it underlines the fact that radiant power is a flow of energy which is emitted by a source and irradiates in all directions in the surrounding space. If the flow is the same in all directions, source is called isotropic. Some sources emit differently according to the direction. Also, the radiant flux can be channeled in a preferential direction with lenses, similarly to the headlights of a car. The value of radiant flux is given by the source, and depends only on the amount of power supplied and not on its spatial distribution.

Energy (J) - a source which has a power of 1 W, emits in one second a unit of energy (1 joule, the symbol being J). For instance, a 60W light bulb supplies 60 J in 1 second when it is turned on. If the lamp is left on for 10 minutes, which is 600 seconds, it supplies 60 W*600s = 36,000 J = 36 kJ. In one hour, it supplies 60W*3,600s = 216,000 J = 216 kJ. Another unit of measurement for energy is the Watt-hour (Wh), which is the amount of energy supplied in one hour by a device which has a power of 1 W; 1 Wh is equal to 3600 J = 3.6 kJ, 1 kWh is equal to 1,000 Wh, which is 3.6 MJ (mega joule, million joule). Therefore both Watt-hour and Kilowatt-hour are a measurement of energy and not power.

Is it best then to think in terms of energy or power? Power is useful if we are interested in knowing the energy which is supplied instantly, as in the case of the light bulb: imagine a 60 W light bulb which is able to supply 60 J in one second (not all the power which is absorbed is turned into light! In fact most of it, at least as far as incandescent light bulbs are concerned, becomes heat.) If instead of 1 second it took 10 seconds to supply the same amount of energy, it would give a much lower light intensity, as much as a 6W bulb. On the other hand, energy is an important measurement for knowing the total amount of energy supplied, as for example in the case of evaluating energy consumption: a light bulb does not "create" energy but transforms it from other forms of energy, such as electricity.

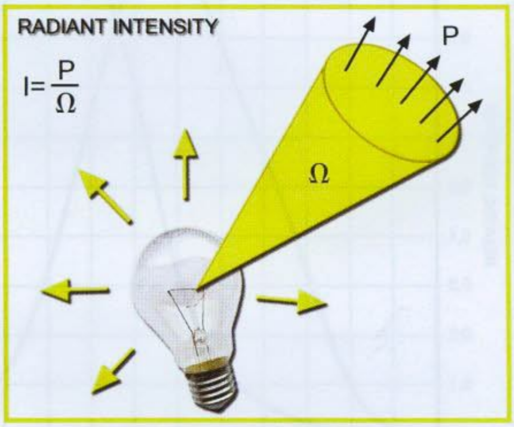
Radiant energy (J) - this is the energy emitted, transferred or received in the shape of radiation. The unit of measurement is the Joule, J = W*s. One can compare it to hydraulic energy: imagine the energy given by a river when it carries out a sudden change in altitude. As this energy can be harnessed by turbines, so can radiant energy emitted by the Sun be gathered by photovoltaic solar panels which transform it into electricity. Another example is photosynthesis, where solar energy is transformed into chemical/biological energy.

Radiant exitance (W/m^2) - it is the radiant flux or radiant power emitted by a source per unit of an area. For example the Sun has a radiant exitance of 1,351 W/m^2, of which only 900 actually reach the surface because of the atmosphere absorbing the rest.

Irradiance (W/m^2) - a synonym for radiant exitance, this represents the radiant flux which hits the surface per area unit. For example the human retina, in normal living conditions, undergoes an irradiance of 10^{-4} W/cm^2, whereas staring at the Sun directly brings irradiance up to 10 W/cm^2, which is 100,000 times more. The following chart indicates irradiance values of a lake at different depths.

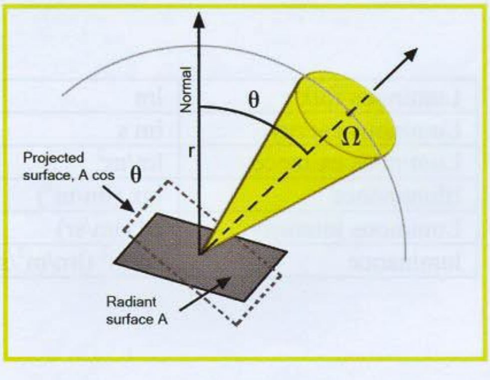
DEPTH (m)	IRRADIANCE (W/m^2)
0	1148.00
-1	1030.00
-2	920.00
-3	723.00
-4	630.00
-6	480.00
-9	381.00

Radiant intensity (W/sr) - by definition radiant intensity is the flow or radiant power emitted by a pointlike source in a given precise direction per unit of a solid angle. As a result, having defined solid angles as being measurable in steradians, the unit of measurement of radiant intensity is Watts/steradians.



■ **Figure 6.125**
Radiant intensity.

Radiance (W/m2/sr) - radiance characterizes the emission of an extended source. It is the radiant flux of an extended source per unit of a solid angle and per area unit projected on a normal plane in one direction. It therefore represents the emission of a non-pointlike source, an extended source in fact.



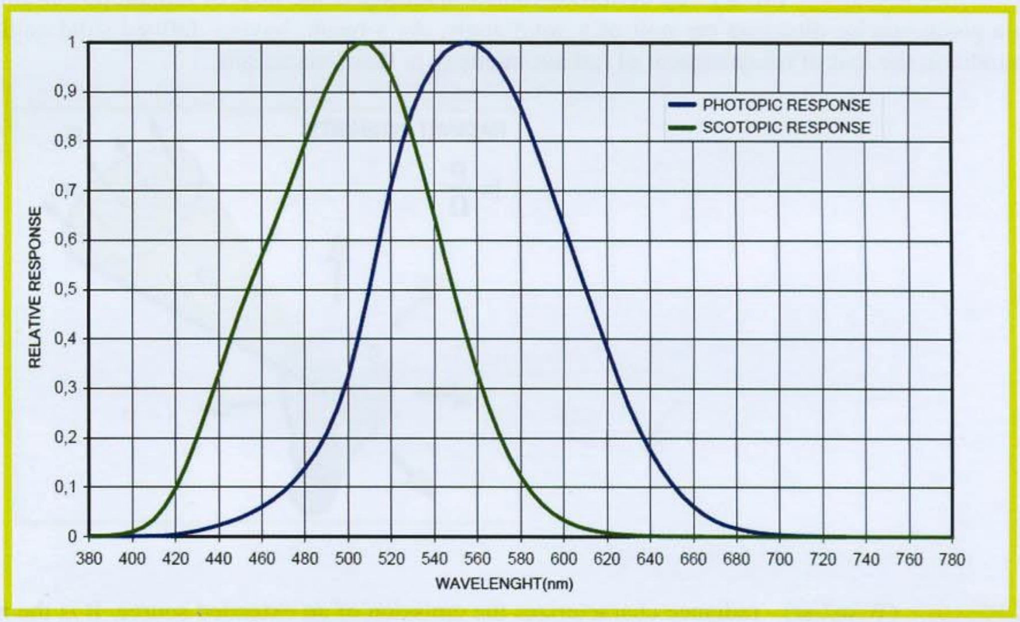
■ **Figure 6.126**
Radiance.

Photometric parameters

The description and measurement of the energy of electromagnetic waves irradiated by light sources uses two types of approaches. The so-called radiometric approach of the physical type is based on the entire radiant power produced by a source, whereas the photometric approach, which is of the psychophysical type, is referred only to the part of the radiant power which is perceived by the human eye as light. The spectrum of radiant energy commonly called light is very narrow, ranging from 300 nm to 750 nm. The shorter wavelengths like ultraviolet and the longer ones like infrared are not perceived by the human eye. As a result there are two types of quantity and units for measurement. The symbols for photometric quantities are the same ones as those used for radiometric quantities.

RADIOMETRIC QUANTITIES			PHOTOMETRIC QUANTITIES		
NAME	*SYMBOL*	*UNIT*	*UNIT*	*SYMBOL*	*NAME*
Radiant energy	Q	J	lm s	Q	Luminous energy
Radiant flux	Φ	W	lm	Φ	Luminous flux
Radiant exitance	M	W/m^2	lm/m^2	M	Luminous exitance
Irradiance	E	W/m^2	lm/m^2 (Lux)	E	Illuminance
Radiant intensity	I	W/sr	lm/sr (Candela)	I	Luminous intensity
Radiance	L	$W/m^2/sr$	$lm/m^2/sr$ (Nit)	L	Luminance

Therefore it is clear that photometric parameters measure quantities which are only related to visible radiation (380 – 780 nm) and weigh them according to the sensibility curve of the human eye.



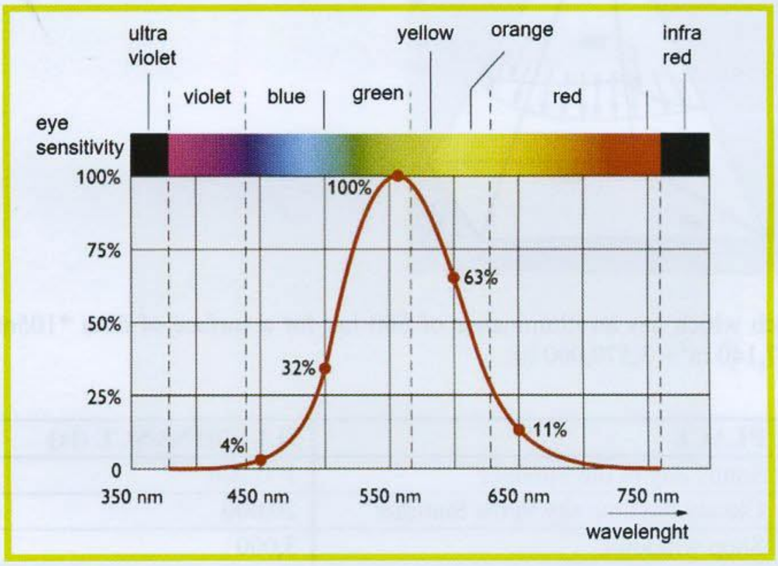
■ Figure 6.127 Graphic representation of the sensitivity of the human eye in photopic vision (with normal illumination levels, like normal daylight) and scotopic (low illumination levels). Illumination at medium levels is called mesopic.

In photopic vision the human eye has a greater level of sensitivity with wavelengths equivalent to 555.02 nm, instead in scotopic of 507 nm.

■ Figure 6.128 The six main photometric quantities.

Luminous flux	lm
Luminous energy	lm s
Luminous exitance	lm/m^2
Illuminance	lux (lm/m^2)
Luminous intensity	cd (lm/sr)
luminance	cd/m^2 ($lm/m^2/sr$)

Luminous flux (lm) - this expresses the quantity of energy emitted by a luminous source per unit of time and therefore define luminous flux as a power (Power = energy/time).
One can compare the luminous flux to a continuous flow of water, measured in liters per second, which sprays in all directions from a tap or a shower. The flux can therefore be thought of as the translation, in terms of light emission, of a power which confirms the energetic nature of light.
When thermal, electric or kinetic energy is restocked by any device which is able to transform absorbed energy into luminous energy, that is, electromagnetic radiations contained within the visible band of the spectrum, a light flux is produced. One could directly employ Watts of absorbed energy to measure the luminous flux. The sensitivity of the visual organ, however, is not constant for all frequencies and wavelengths. The human eye responds to stimulation provoked by radiation with a chain of maximum intensity nervous impulses only when λ values are contained in a narrow interval. The impulses decrease in intensity gradually as one draws further away from this interval. It would therefore be incorrect to consider the power emitted by different wavelengths in the same way. One could say that for the eye a part of the flux is more important than the rest. This is the reason why, when calculating the entity of the luminous flux generated by a source, one takes into account the characteristic sensitivity of the human eye.
In order to obtain a connection between radiometric and photometric quantities, one defines the lumen (lm) as the photometric unit of a luminous flux, in such a way that 1 Watt of radiation emitted at 555.02 nm produces a luminous flux of 683 lumen. 1 Watt of radiation emitted at different wavelengths produces less than 683 lumen, as the human eye has its maximum sensitivity peak at 555.02 nm.



■ **Figure 6.129**
The sensitivity scheme of the eye in photopic vision.

Take the example of a lamp which emits 2 Watts of radiation at 555 nm and 3 at 820 nm. How many lumens are emitted by the lamp? The 2 Watts emitted at 555 nm produce 683 lm each. So we have 2*683 = 1366. The 3 Watts emitted at 820 nm do not produce any lumens at all, as the eye has no reception at that wavelength. So the total number of lumens produced is 1,366 lm.

■ **Figure 6.130**
The flux is the basic unit of luminous power. If it is expressed in Watts, it concerns radiometry. If expressed in lumens it concerns photometry.

TYPES OF LAMPS	ABSORBED POWER	LUMEN
Bicycle light	2 Watt	18 lm
Incandescent lamp	40 Watt	350 lm
Incandescent lamp	200 Watt	3000 lm
Fluorescent lamp	40 Watt	2500 lm
Mercury vapor lamp	400 Watt	2300 lm
High pressure lamp	400 Watt	38000 lm

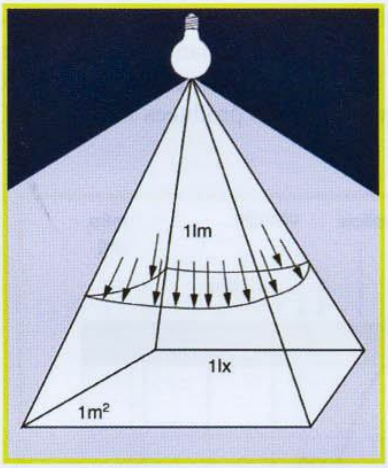
■ **Figure 6.131**
Values of luminous fluxes given by some types of lamps.

Light energy (lm*s) - it is sometimes also called light quantity and stands for the part of radiant energy perceived as light per unit of time.

Luminous exitance (lm/m^2) - when considering the characteristics of a light source, one must specify how much energy is emitted in a given portion of space. If one considers a radiant (or rather, luminous) surface, one uses the luminous exitance parameter and one measures it in lm/m^2.

Illuminance Lux (lm/m^2) - in many fields and applications, one is more interested in determining the flux that hits a surface than the emitted flux. This quantity is equivalent to the exitance we have just described and is called Illuminance. The difference between Illuminance and exitance lies in the fact of considering the energy which is incident as opposed to that which is being irradiated. The unit of measurement is called Lux (lx) and is equivalent to 1 lm/m^2.

■ **Figure 6.132**
Graphic representation of illuminance.



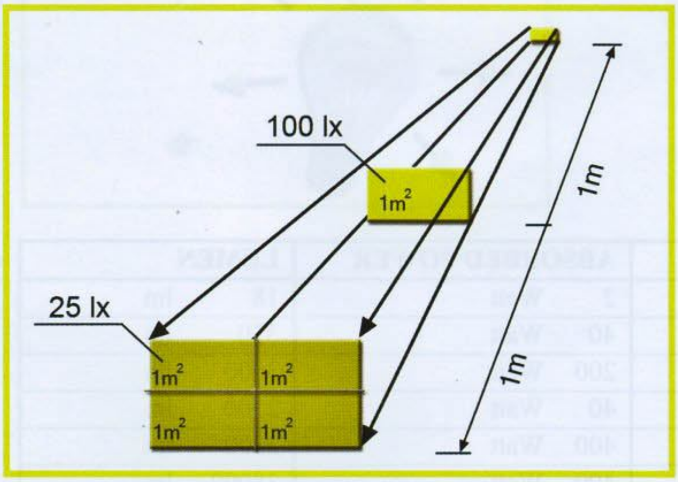
For example, a football pitch which has an illuminance of 500 lux for a surface of 68m *105m = 7140m^2 receives a luminous flux of 500 lux * 7,140 m^2 = 3,570,000 lm.

■ **Figure 6.133**
Typical illuminance values.

PLACE	ILLUMINANCE (lx)
Sunny day in the Summer	100,000
Cloudy daytime sky in the Summer	20,000
Shop windows	3,000
Offices	500
Dining halls	200
Streets at night	30
Night with full Moon	0.25
Night with clear sky and no Moon	0.01

Lastly, we will make a consideration regarding geometry: illuminance varies following the inverse of the square distance of the light source.

■ **Figure 6.134**
Inverse square law.

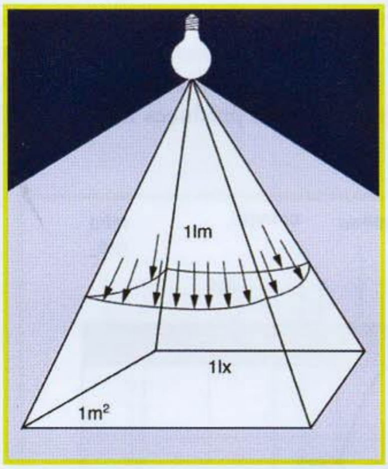


Light energy (lm*s) - it is sometimes also called light quantity and stands for the part of radiant energy perceived as light per unit of time.

Luminous exitance (lm/m^2) - when considering the characteristics of a light source, one must specify how much energy is emitted in a given portion of space. If one considers a radiant (or rather, luminous) surface, one uses the luminous exitance parameter and one measures it in lm/m^2.

Illuminance Lux (lm/m^2) - in many fields and applications, one is more interested in determining the flux that hits a surface than the emitted flux. This quantity is equivalent to the exitance we have just described and is called Illuminance. The difference between Illuminance and exitance lies in the fact of considering the energy which is incident as opposed to that which is being irradiated. The unit of measurement is called Lux (lx) and is equivalent to 1 lm/m^2.

■ Figure 6.132
Graphic representation of illuminance.



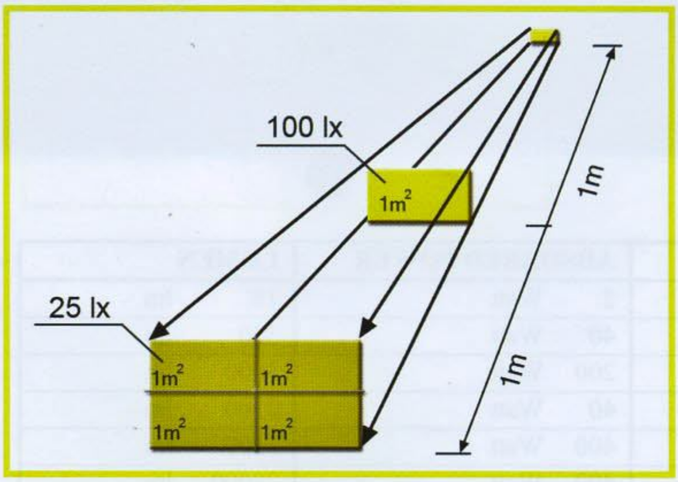
For example, a football pitch which has an illuminance of 500 lux for a surface of 68m *105m = 7140m^2 receives a luminous flux of 500 lux * 7,140 m^2 = 3,570,000 lm.

■ Figure 6.133
Typical illuminance values.

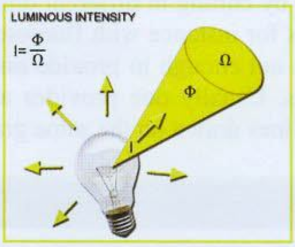
PLACE	ILLUMINANCE (lx)
Sunny day in the Summer	100,000
Cloudy daytime sky in the Summer	20,000
Shop windows	3,000
Offices	500
Dining halls	200
Streets at night	30
Night with full Moon	0.25
Night with clear sky and no Moon	0.01

Lastly, we will make a consideration regarding geometry: illuminance varies following the inverse of the square distance of the light source.

■ Figure 6.134
Inverse square law.



Luminous intensity (cd = (lm/sr)) - by definition luminous intensity is the part of luminous flux emitted in a certain direction by a light source within the solid angle which contains it, in other words it measures the luminous flux Φ emitted through the solid angle Ω.

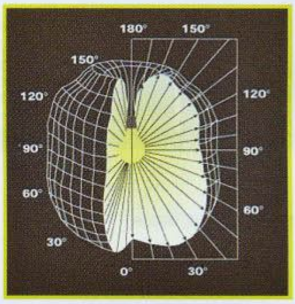


■ **Figure 6.135**
The candela is one of the seven units of measurement in the International System of units of measurement.

An ideal pointlike luminous light source irradiates its luminous flux in all directions in space, which means its luminous intensity is the same in all directions. However, in reality one obtains a heterogeneous spatial distribution of the luminous flux, partly because of the structure of a lamp, partly because of the effect of the direction it is aimed in.
Luminous intensity defines the quantity of light which is given off by a light source in a certain direction and is measured in candelas (cd). Luminous intensity is a quantity defined as vectorial. To indicate it, it is not sufficient to give a quantity, a number. One must also express the direction associated with that number. So one can talk about intensity only when a direction is indicated. In illumination engineering one expresses the direction by giving an angle compared to the vertical of the light source. With the usual hydraulic analogy, luminous intensity can be compared to the pressure of a jet of water in a certain direction. Intensity not only tells us how much light comes from a source but above all what its direction is. Because of this, intensity is an extremely useful parameter for measuring objects which give off light.

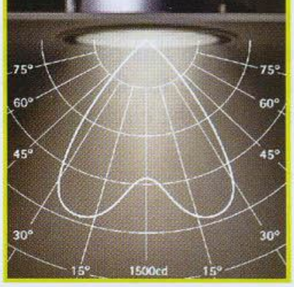
LUMINOUS DEVICE	CANDELAS A (cd)
Led	0.0005
Candle	1
100 Watt incandescent light bulbs	150
Car headlights	100,000
Lighthouse	300,000

The spatial distribution of the luminous intensity of a source produces a three-dimensional web of distribution of the this luminous intensity. If one chooses to represent the intensities given off by the device in the surrounding space with a series of vectors coming from the center of the light source, the zone of the extremes of these vectors is a surface which encloses a volume, called "photometric solid".



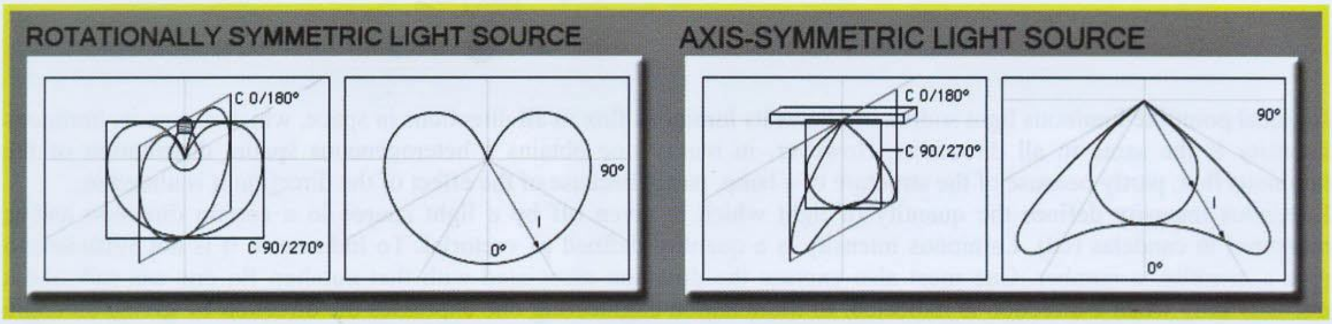
■ **Figure 6.136**
Example of a photometric solid.

A luminous body emits light in all the directions in space. If for each direction in space one represents the intensity vector as a segment coming out of the source, with a length proportional to the value of intensity, one is left with lots of "needles" which together represent the photometric solid. The photometric curve is nothing more than a cross section of the photometric solid.



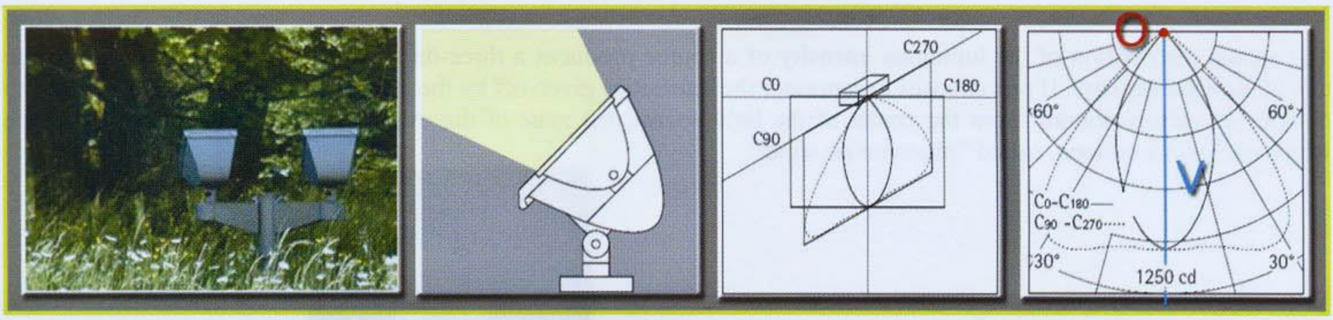
■ **Figure 6.137**
Example of a photometric curve.

It is as if the photometric solid were an apple and the photometric curve shows a side view of the apple after it has been cut in half. Often luminous bodies have a symmetric emission, and the photometric solid has a shape which is really quite similar to that of an apple. Any of its "slices" has the same profile. If this is the case, it is enough to have the curve of a single slice in order to have a complete representation of the whole device. In other cases, the solid can have a more irregular and jagged shape: by cutting in different directions, "slices" are completely different, more similar to a potato than an apple. This happens for instance with fluorescent lights, which have one of the two dimensions bigger than the other. In this situation, it is not enough to provide one curve; instead one must create many of them by sectioning the solid along different planes. Usually one provides at least two, relative to two planes, perpendicular to each other, shown by two different outlines drawn on the same graphical drawing.



■ **Figure 6.138**
Different curves for light apparatus.

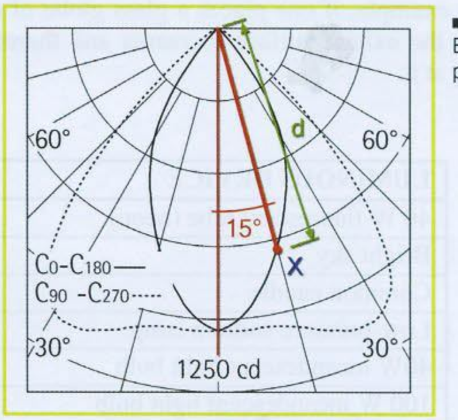
Reading a photometric curve can be difficult if one has never done so before and is not used to polar diagrams. Apart from their technical meaning, photometric curves are quickly understandable and in a quick glance one can grasp the way a device works. With a little practice one can easily distinguish in very little time the degree of concentration of the beam of light that a device emits, whether it has double emission, is symmetric or not and so on. Polar diagrams are graphic schemes which show a portion of plane with a center, the origin, and a reference axis which starts from the centre.



■ **Figure 6.139**
Real example taken from a catalogue of light appliance.

Any point of the plane can be found, simply by indicating the angle in relation with the reference axis V and the distance from the origin. In order to make it easier to read these measurements, usually polar diagrams have concentric circles surrounding the O origin and lines coming from the origin which divide the plane into slices. Each illumination device has its own photometric curve. The origin of a polar diagram represents the point where the illuminating object is located and the reference axis represents the vertical of the device. In order to read the intensity values associated to each direction on the graphic drawing, one must proceed in the following way:

. One finds the angle of interest, one traces a line coming from the origin in that direction. In the example it is 15°.
. One finds the point where the line meets the outline of the photometric curve: the X point.
. One measures the distance between this point and the origin (d). This distance, in relation to the scale of the curve, represents the intensity of the device for the angle one has decided to analyze. It is easily readable on the graduated scale shown in correspondence with the circles centered on the origin.



■ **Figure 6.140**
Example of the reading of a photometric curve.

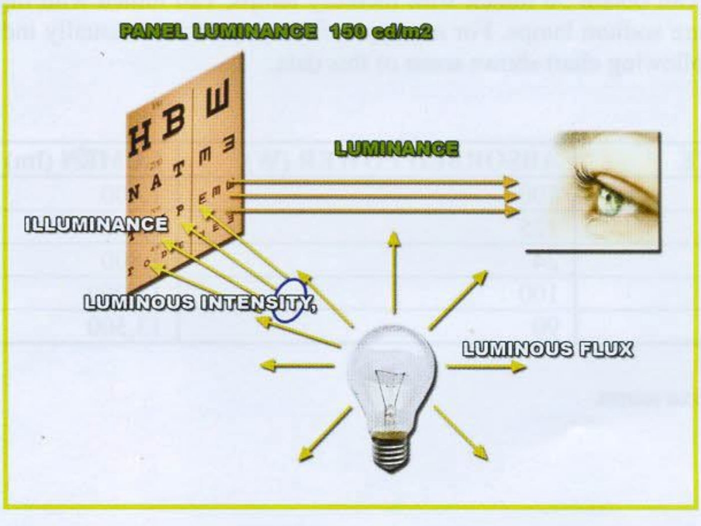
At this stage the problem that arises is how to quantify luminous intensity. Intensity is measured in candelas: therefore photometric curves are used for reading values in candelas. However, often photometric curves are not expressed in absolute values, but in candelas over lumen (cd/lm). In many cases, typically in fluorescent devices, the element has a certain intensity distribution which depends on its constructive characteristics (type of shielding) regardless of the lamp which it is placed in. Therefore the shape of the photometric curve is defined. However, it may happen that in the same device many different lamps are attached, each one corresponding to its own luminous flux. Different lamps may correspond to the same shape but a different size or scale. In these cases it is best to express only the shape of the curve and leave the user a relative value with which he can calculate the real absolute value of intensity according to the lamp which is being used. Photometric curves expressed in cd/lm show how many candelas are emitted for every 1,000 lumens of flux of the lamp divided by 1,000. For example:

Lamp in use = … 5,400 lm
Read value = ……….. 25,000 cd/lm

Actual absolute value = 25,000 * 5,400/1,000 = 25000 * 5.4 = 135,000 cd

The unit of measurement which has been employed, whether in cd or cd/lm, is always shown in the graphic drawing of curves: one must take care not to get absolute values (cd) mixed up with relative values (cd/lm).

Luminance (cd/m2 = lm/m2/sr) - luminance is defined as the ratio between luminous intensity emitted by a source towards a surface, normal compared to the direction of the flux, and the area of the surface itself. It is expressed in cd/m^2.
This quantity indicates the dazzling quality that a source can induce. For example a small size lamp with an emission surface of one square centimeter with the intensity of one candela, has a luminance equivalent to 10,000 cd/m^2. Another lamp with the same intensity but with a surface of one hundred square centimeters has a luminance equal to 100 cd/m^2. One can see why it is much less annoying to observe a fluorescent tube, which has a wide surface, rather than an incandescent light bulb, where all the light flux comes from the small filament. This enables us to comprehend that it is a parameter which takes into account the surface of irradiance of the light source.



■ **Figure 6.141**
Summary example of some photometric parameters.

This is what happens, for example, if one places a glass globe of opaline around a naked lamp or a candle: the whole globe becomes luminous; the exitant surface increases and therefore luminance decreases, and with it the annoying sensation one gets looking at it.

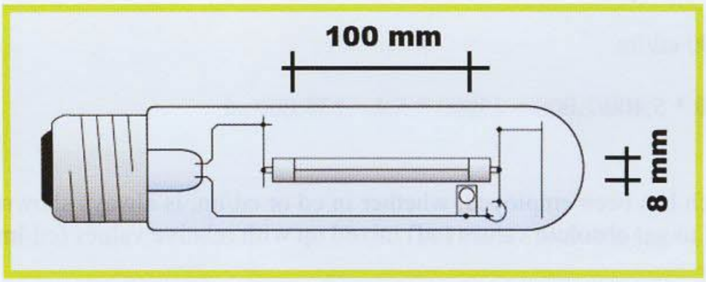
■ **Figure 6.142**
Luminance of a few luminous sources.

LUMINOUS DEVICE	LUMINANCE (cd/m2)
40 W fluorescent tube (neon)	7,000 - 15,000
Bright sky	4,000
Common candle	5,000
Low-pressure sodium lamp	20,000
40W incandescent light bulb	5,000,000
100 W incandescent light bulb	6,000,000
1,000 W incandescent light bulb	12,000,000
Medium-pressure mercury vapor bulb	3,000,000 - 10,000,000
High-pressure mercury vapor bulb	300,000,000 - 1,700,000,000
The Sun	1,650,000,000

For a calculation example, imagine a sodium lamp, like the one in the image. The discharge tube measures 8mm*100mm and emits, perpendicularly to the cylindrical surface, a luminous intensity of 4,000 candelas. The luminance of this surface for an observer placed in the same direction is:

$L = cd/m^2 = 4{,}000 / (100*8) = 5\ cd/mm^2$ o $5{,}000{,}000\ cd/m^2$

■ **Figure 6.143**
Scheme.



Luminous efficiency (lm/W) - the luminous efficiency of a light source is the ratio between the luminous flux of a source and the total energy flux (power) emitted by the source. The luminous flux is defined according to the subjective perception of the average human eye and corresponds to a particular curve within the spectrum of visible light. A light bulb emits radiation also outside of the visible band, usually in infrared and ultraviolet areas, which however cannot contribute towards the visual sensation of brightness. A lamp has a greater luminous efficiency the more it is able to emit a spectrum which is appropriate to human perception. The unit used for measuring is lumen/Watt.
Each lamp has its own luminous efficiency, that is, it produces a certain amount of light for a given absorbed power. By replacing a scarcely efficient lamp with a more efficient one, one obtains, at equal power, a greater amount of light. For example, with 1 Watt one can obtain 50 lumen with mercury lamps, 120 lumen with high-pressure sodium lamps and 150 lumen with low-pressure sodium lamps. For any type of lamp, producers usually indicate luminous flux as well as power consumption. The following chart shows some of this data.

LUMINOUS APPLIANCE	ABSORBED POWER (W)	LUMEN (lm)	EFFICIENCY (lm/Watt)
Incandescent light bulb	100	1,400	14
Mercury vapor	125	6,300	50
Fluorescent (neon)	24	1,800	75
High-pressure sodium	100	12,000	120
Low-pressure sodium	90	13,500	150

■ **Figure 6.144**
Luminous efficiency for a few luminous sources.

The mechanical equivalent of light is equal to 621 lm/W, so if all the electric energy absorbed by the lamp was transformed into luminous flux in the unit of time, the power of 1 Watt would give 621 lm. From the chart one can confirm that a lamp with a luminous efficiency of 14 lm/Watt and a power of 100 W emits 1,400 lumen, against a theoretical 62,100 lm/Watt, with an efficiency of 2% and luminosity loss of 98%.

Efficiency = lm/Watt

The result is an adimensional number. The ratio, in fact, takes place between two quantities of the same unit, two power values. With the inverse formula one can obtain the lumen value; in other words the part of energy transformed into light.

At this stage, it is necessary to know, at least in general, the main types of light sources offered on the market. This is useful for better understanding terminology, such as incandescence, halogen and discharge light bulbs.

FILAMENT BASED SOURCES

These are the oldest and unfortunately still the most widespread electrical appliance for producing illumination. The electric current is lead through a thin metal filament, usually made in tungsten, placed within a glass globe, where a vacuum has been created or filled with an inert gas, in order to avoid the oxygen from burning the filament. The passage of electric current induces the heating of the filament, which becomes incandescent, and can reach a heat of 2,500/2,700°C, emitting both light and heat.

Advantages of incandescent light bulbs:

- they turn on immediately
- warm light color
- immediate reaching of maximum efficiency

Disadvantages of incandescent light bulbs:

- Scarce luminous efficiency (max 20, 25 lm/W)
- Extremely delicate
- Low durability

Traditional incandescent light bulbs.

These are the traditional light bulbs which can be bought in a wide range of formats and power variation. The main advantages are the fact of being able to control the luminous flux, the speed of ignition and deactivation and a good chromatic quality, On the other hand they are scarcely efficient, (10% of the energy is converted to light in the best of cases, the rest is transformed in heat) and they have a short life, at the most 1,000 hours.

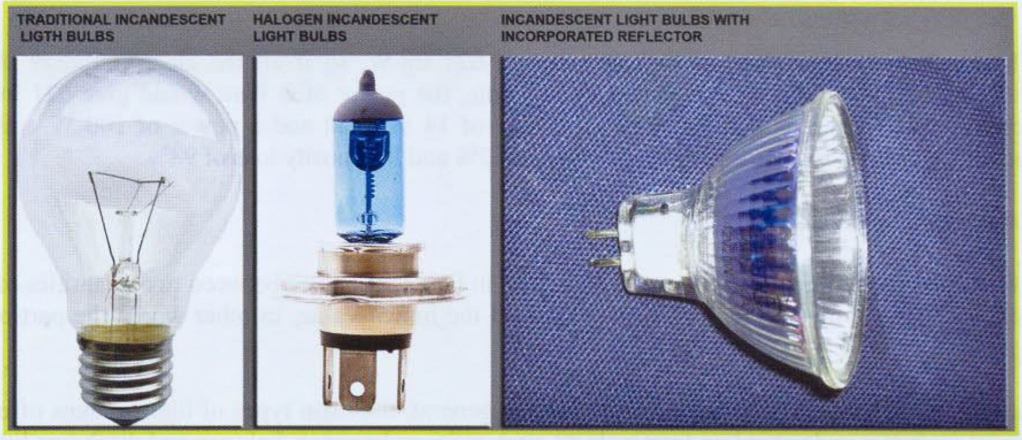
Halogen incandescent light bulbs.

These are similar to the above, except that they have small quantities of halogens, such as iodium or brome within the glass globe. The result is that they last longer (2,000 to 3,000 hours), a whiter light with excellent chromatic return, the possibility of working with low voltage with the help of a transformer. On the downside, efficiency is still very low, surface temperature is very high, whereas the use of quartz in the manufacturing process of the bulbs produces an increase of ultraviolet rays.

Incandescent light bulbs with incorporated reflector

Similar to the previous type, but projected to give a beam of light in a specific direction. The lasting lifespan of these lamps is at least twice as much as normal ones. According to the type of reflective material and the shape of the light bulb, the amount of light projected by the beam can increase, at equal Watt power, from 50% to 300% compared to the average light bulb.

■ **Figure 6.145**
Three types of incandescent light bulbs.



DISCHARGE LIGHT BULBS

In this type of bulb one exploits the property of certain types of gas of being able to produce luminescence when an electric charge goes through them. In a glass or quartz tube vacuum is produced and then a small amount of gas or metallic vapor is inserted: two electrodes placed on the extremes of the tube produce an electric arc which induces ionization of gas and this gives way to a process which transforms energy fed into it into visible radiation.

Low-pressure mercury-vapor lamps

The fluorescent lamp is a particular type of discharge bulb where visible light emission is indirect, that is to say, it is not emitted directly by the ionized gas, but by a fluorescent material, hence the name. It is made of a tube of glass which can be linear, circular or molded, within which first of all vacuum is produced, then a noble gas is inserted, such as argon or neon, and a small quantity of liquid mercury which is balanced with its vapor. The internal surface of the tube is covered with a fluorescent material, which looks like a white powder. At the two extremes of the tube, two electrodes are placed. The electrons which move between the two electrodes excite the mercury atoms which are in turn stimulated to produce an ultraviolet radiation. The phosphor which dresses the tube, when hit by these radiations, emits visible light. A different composition of fluorescent material allows colder or warmer tonalities of light to be produced.

Ignition times are quite quick, although not immediate like filament bulbs. The luminous flux can be regulated, thanks to the help of electronic auxiliary units such as condensers, starters and so on. Efficiency is high, lifespan reaches up to 8,000 hours and luminosity is very strong. The quality of light and the chromatic efficiency depend on the blend of phosphors used and can vary greatly, from mediocre to excellent.

Advantages of low-pressure mercury vapor lamps

- Good luminous efficiency which can reach 90 lm/W.
- Small size.
- High reliability.
- Long lasting.
- They work in any position.
- Low cost.

Disadvantages low-pressure mercury vapor lamps:

- 4 or 5 minutes before full operativity is reached.
- Power peak at ignition.
- Difficulty in disposal (mercury is a special type of waste).

Metallic halogen lamps

Lamps with metallic halogen vapors are a family of light sources which is very vast and varies greatly for different sizes, colors, power range and electric standards. In fact this technology allows one to obtain very different effects, by differentiating the types and quantities of metallic compounds which are contained in the tube. Up to today there is no standardization between manufacturers, who have all chosen different paths, some persuing aspects of durability and reliability and others more interested in characteristics concerning color and chromatic efficiency. Light sources based on discharge of the metallic halogen vapor type have reached standards of quality which are particularly high. The recent evolution of technology concerns mainly the tube itself. A translucent ceramic material (polycrystalline alluminium), already having been successfully experimented on other types of discharge sources (high-pressure sodium vapor types) has been adopted. This innovation allows one to manufacture reliable lamps with light tones which remain constant in time (at color temperatures of 3000 K), in any position they are used in and regardless of variations in energy supply voltage. Efficiency varies from 89 lm/W to 92 lm/W. Color temperature corresponds to 3000 K, with warm light tones.

Advantages of halogen lamps:

- Excellent chromatic efficiency.
- High luminous efficiency.

Disadvantages of halogen lamps:

- Short durability (about half) compared to mercury vapor types.
- Limited functioning positions.
- High price.
- High re-ignition time when light is still warm from use (10 – 15 min).

Sodium vapor lamps

They function in exactly the same way as mercury vapor types. The difference is in the bulbs, which are filled with sodium. There are high-pressure and low-pressure versions.
The low-pressure sodium lamp was the first gas-based discharge bulb introduced in 1932 and still today remains the best light source, as far as efficiency is concerned. This is the reason why low-pressure sodium lamps are still used, in spite of their low quality monochromatic light with only one band of emission around 600 nm, which is the area of maximum sensitivity for the human eye. They are used mainly for road-side, industry and security installations. Another interesting feature of these lights, is that they are environmentally friendly, as they do not contain mercury. There is little to say about chromatic efficiency, as the perception of colors in this light is practically none. By increasing the pressure, the sodium vapor draws further away from the state of ideal gas and its emission spectrum grows wider compared to its typical spectral range. The light produced by these light bulbs is white and tends towards yellow (2000 – 2500 K), a characteristic which makes them unfavorable for activities where color efficiency is important. Light returned on power is high (190 Lumen/Watt) and also durability (over 15,000 hours). Recent bulbs with high-pressure white sodium offer a realistic color perception, unlike the first sodium light bulbs, which provided an almost monochromatic yellow light. Ignition and re-ignition times are long and intensity cannot be regulated: to make up for this, these are certainly the most efficient bulbs around at the moment.

Advantages of high-pressure sodium lamps:

- High luminous efficiency.
- Long durability.
- Acceptable chromatic efficiency.
- Small size.
- Correct functionality in any position.

Disadvantages of high-pressure sodium lamps:

- Voltage variation can lower duration.
- Full working potential is reached after 5 or 6 minutes.
- Power peak at ignition.
- High costs compared to fluorescent mercury vapor bulbs.

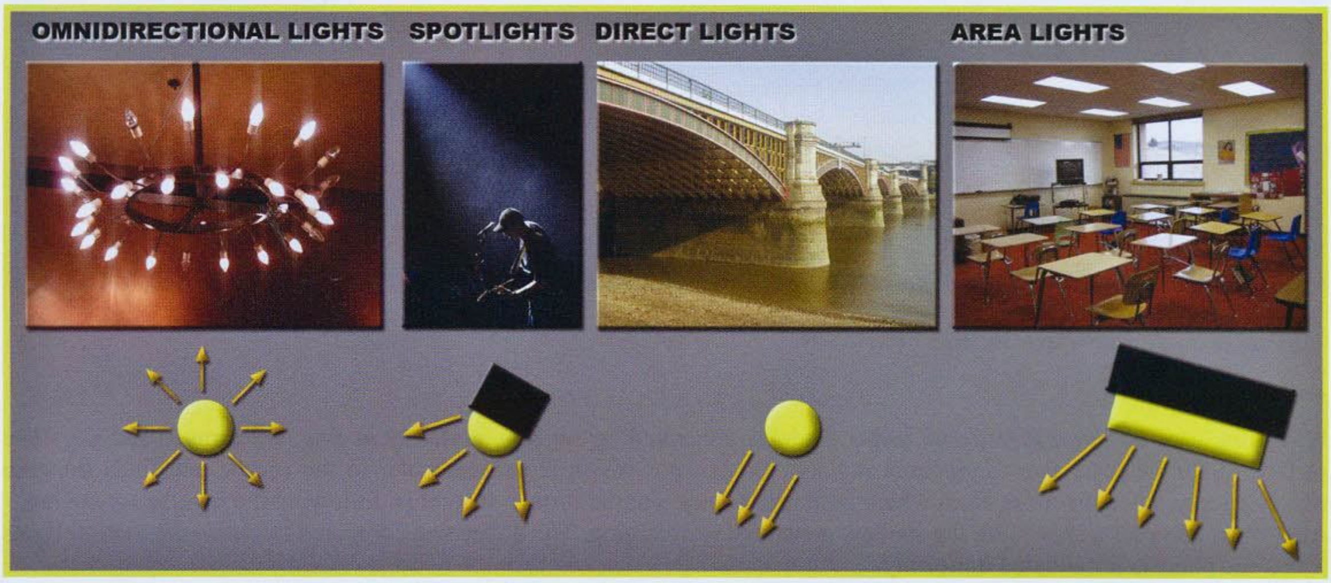
■ **Figure 6.146**
A few types of discharge light bulbs.

The following is a small summarizing chart with data concerning different light sources.

LIGHT SOURCE	EFFICIENCY	DURATION (in hours)	COLOR TEMP °K
Incandescent light bulbs	14	1,000	2,750
Halogen light bulbs (Tungsten)	20	2,000	2,850
Fluorescent bulbs	100	12,000	2,700-6,000
Metallic halogen bulbs	80	300-1,000	5,500
High-pressure sodium bulbs	130	9,000	2,000
Low-pressure sodium bulbs	190	10,000	1,800

3ds max lights

Lights are objects which can simulate light sources, such as household or office lamps, stand reflectors or cinema reflectors, the Sun and so on. Within 3ds Max there are different types of light sources, which emit light in different ways and simulate most types of real light sources.



■ Figure 6.147
Main types of light sources.

In everyday life there are four main categories of light sources:

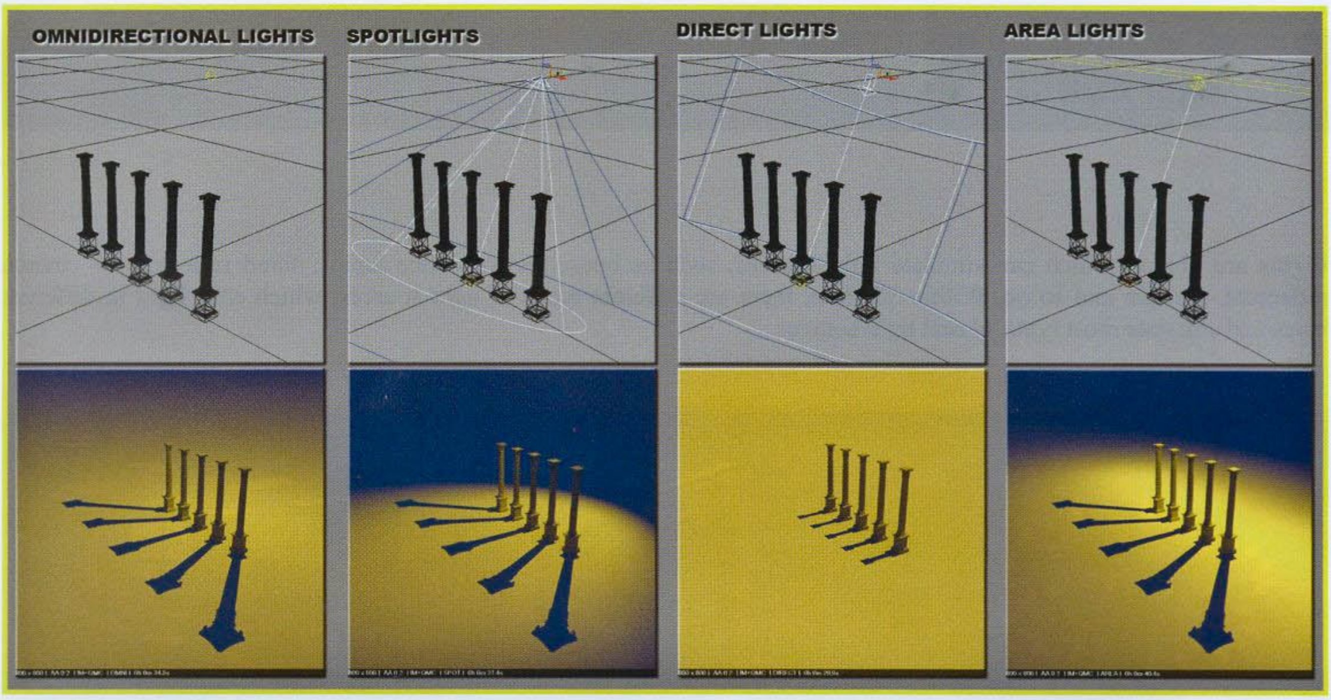
. **Omni Lights:** the luminous flux is emitted uniformly in every direction, like bulbs without shielding. Light rays irradiate from a small point radially.

. **Spotlights:** the luminous flux is directional and illuminates a small portion of space around the light source, because it is shielded. Like the previous sources, also in this case rays are not parallel but diverging.

. **Direct lights:** this is the specific case of the Sun. As it is extremely distant from the earth, rays can be considered parallel to each other by the time they reach the earth.

. **Area lights:** the luminous flux, in this case, is emitted by a surface which has a well-defined and visible area. Rays are diverging, as they are in spotlights and omni lights.

Knowing the behavior of rays for a light source is important especially for shadow generation and surface illumination.



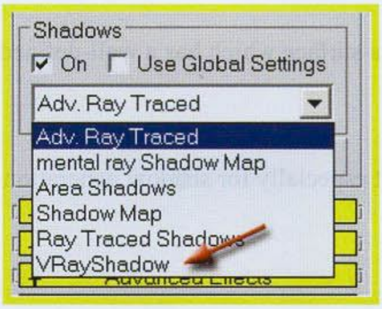
■ **Figure 6.148**
The main illumination systems within 3ds Max.

When one uses an ***omni-directional*** light, shadows generated by objects, perfectly in line, are arrayed radially. The surface of the plane is also illuminated radially. In the spot which is perpendicular to the source one finds the area which is illuminated the most, from which light tends to die out the further away it is, following the inverse square law.
Also when one uses a ***spot*** light, the shadow generated is diverging. The difference compared to omni- directional lights is that the plane is illuminated uniformly.
With ***direct*** light the situation changes completely. The plane is illuminated homogeneously and simulates the behavior of the Sun. Also shadows are perfectly parallel.
Lastly we come to ***area*** lights. In this case the shape of the source light has been defined in such a way that it is long and thin, simulating a neon light. Unlike ***spot*** lights or ***omni-directional*** lights, the plane is illuminated as if it were being lit up by a long fluorescent light. The orange gleam under the source is in fact rectangular. In this case, shadows are distributed radially.

VRay supports almost all the light sources. The ones which are not compatible are the lights in Mental Ray, the 3ds Max skylight, and the IES Sky, IES Sun, Daylight, Sunlight. There is no need to worry about these last five types not being supported, as *VRay* version 1.5 has replaced them with the system ***VRaySky+VRaySun***.

Another characteristic of shadows, as well as the way they are aligned (radially or in a linear manner), is the way they are calculated. 3ds Max provides various classes of shadows, each one with its own characteristics.

■ **Figure 6.149**
The five different types of shadows in 3ds Max. VRay adds a one type of its own.



If we leave out the *VRay* shadows temporarily, shadows can be classified as:

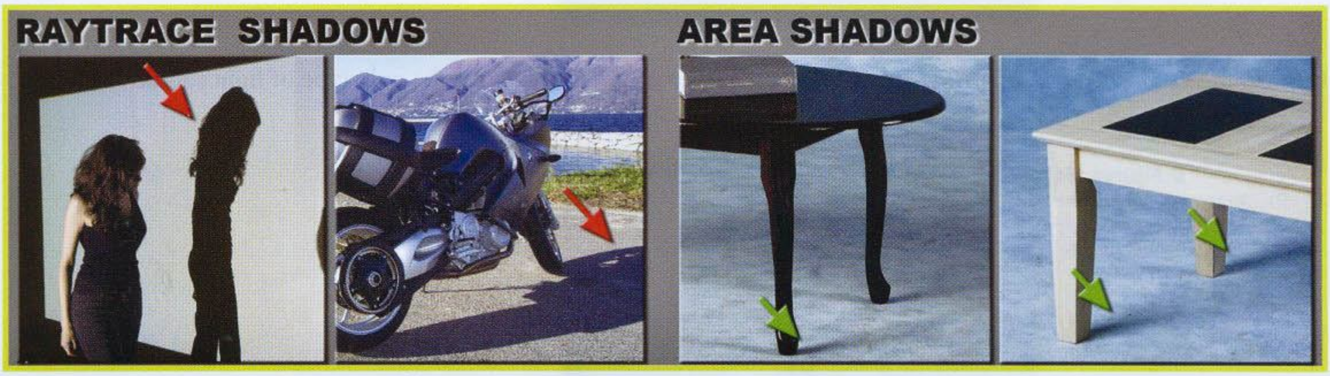
Shadows Map: shadows, in this mode, are managed and generated as if they were maps. It is in fact possible to define the quality as if it was a texture, by setting their resolution. They are faster in calculation compared to other types. When there are lots of high-definition lights, however, they consume large amounts of memory. They can generate both smoother or sharper shadows, but nowhere near the sharpness and precision of Raytrace shadows.

Raytrace: their peculiarity lies in their extreme precision and quality, but they are also slow. Shadows produced with this method have very defined outlines, which cannot be obtained in real life. Their appearance is very cold, too much so, and typically associated with "computer-generated images", because of their high precision.

Area Shadows: they allow one to generate photorealistic shadows and hence create smooth shadows. As is the case with real-life light sources, this type of method produces shadows which are not very sharp. The further away shadows are from the object, the more it will tend to open up like a fan, losing definition. The amount of blurring depends mainly on the distance of the source from the illuminated model, but also on the shape and size of the light source. These shadows have a very pleasant and real-looking appearance. Unfortunately they are slow to be calculated.

■ **Figure 6.150**
The three different types of shadows.

VRay does not support, within 3ds Max, anything which involves Raytrace calculations, like for example Raytrace materials, but also Raytrace Shadows themselves. For this reason, neither Raytrace shadows nor Area Shadows are accepted. However, ***Chaosgroup*** has made available the proprietary lights of 3ds Max, a type of shadows which allow one to obtain both Raytrace shadows with defined borders and Area Shadows with blurriness: it is the case of ***VRayShadows***.



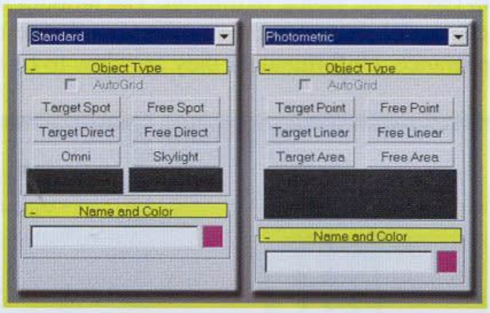
■ **Figure 6.151**
Raytrace and ***Area*** shadows in real-life.

As far as the next studies and examples are concerned, we shall always use ***VRayShadows***.

VRayShadow

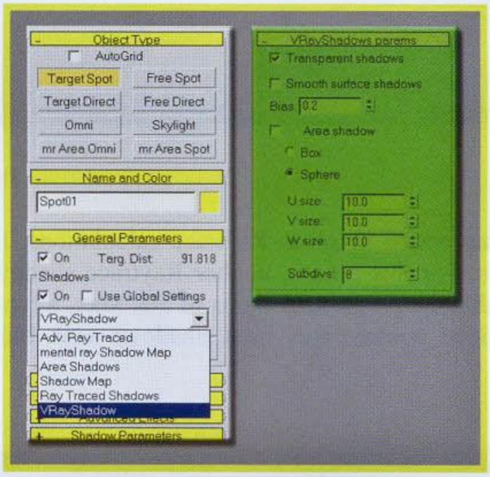
INTRODUCTION

VRayShadows are a type of shadow specifically studied to be used in ***VRay***. They are used for generating Raytrace shadows, as a replacement for 3ds Max Area Shadows and Raytrace Shadows.



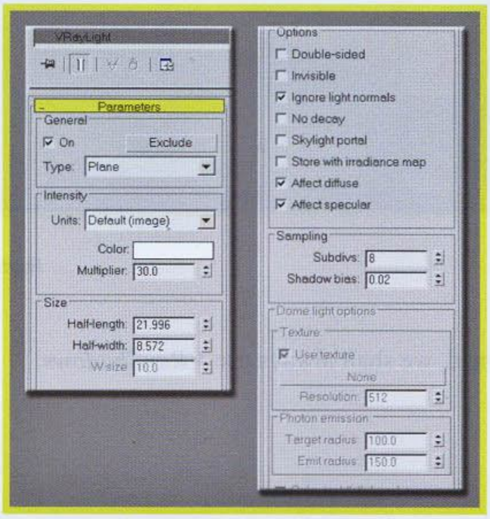
■ **Figure 6.152**
Light sources supported by VRay.

VRayShadows can be found in the General Parameters rollout in the 3ds Max lights section. In order to activate them, it is sufficient to select ***VRayShadows*** in the pull-down menu. Automatically, a new rollout appears, called ***VRayShadows params***.



■ **Figure 6.153**
VRayShadows are located within the 3ds Max lights area. In this specific case we are looking at a Spot Light.

VRayShadows are also part of ***VRayLights***, which, together with ***VRaySun***, are the only type of light source in ***VRay***.

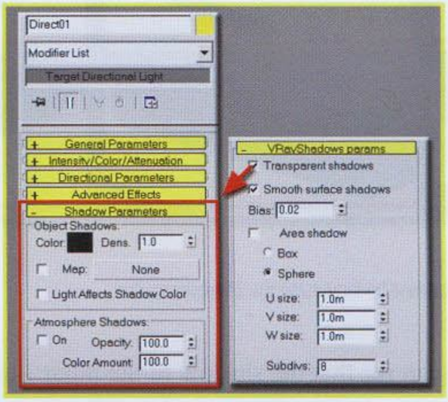


■ **Figure 6.154**
The VRay lights panel: *VRayLight*. All the features of *VRayShadows* are contained within *VRayLight*.

In order to use ***VRayShadow*** together with 3ds Max lights one must create the type of lights desired in the viewport, as one would do with any 3ds Max light, activate the shadows and choose ***VRayShadows*** from the pull-down menu. At this stage everything is ready to be used and finally it is possible to generate sharp or smooth shadows, called ***Area shadows*** in jargon.

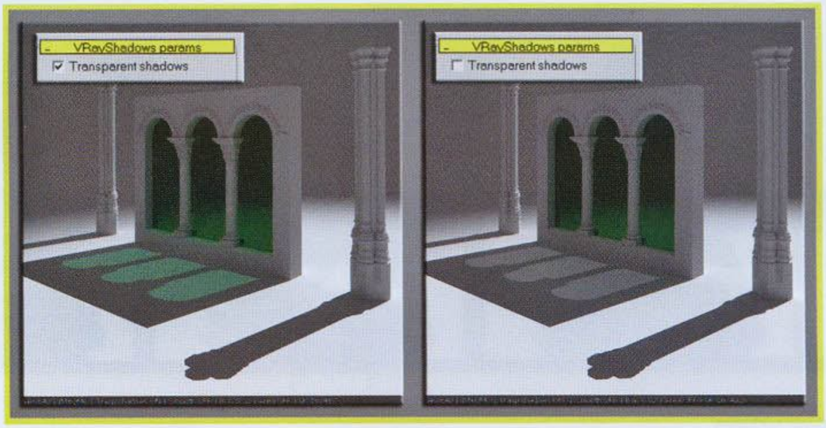
PARAMETERS

Trasparent shadows - this parameter determines the behavior of shadows when there are transparent or semi-transparent objects. If this option is activated, *VRay* calculates the shadow produced by the transparent object, regardless of parameters present in the Shadow Parameters rollout, such as Color, Density, Map and so on.



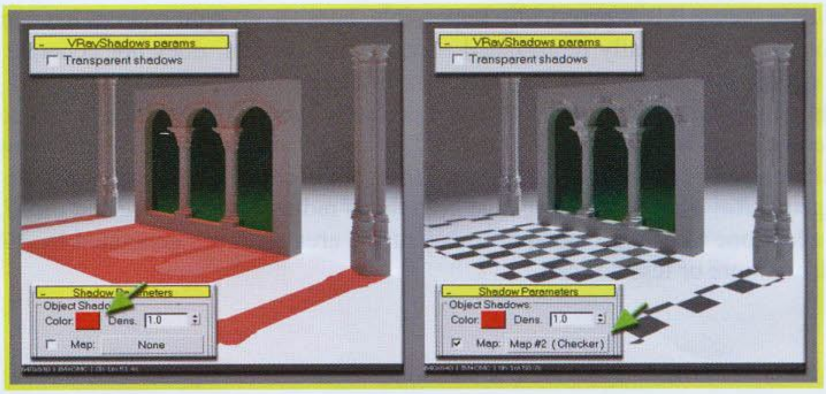
■ **Figure 6.155**
The parameters of the ***Shadow Parameters*** rollout can be used only if the ***Transparent Shadows*** option is deactivated.

As a result objects produce a colored shadow which corresponds to the color of the material, specifically a ***VRayMtl***. If ***Transparent shadows*** is deactivated, the parameters in the ***Shadow Parameters*** rollout will be taken into account and the shadow generated by semi-transparent objects will be monochromatic.



■ **Figure 6.156**
Notice how the shadow of the glass pane is correctly colored when ***Transparent shadows*** is activated.

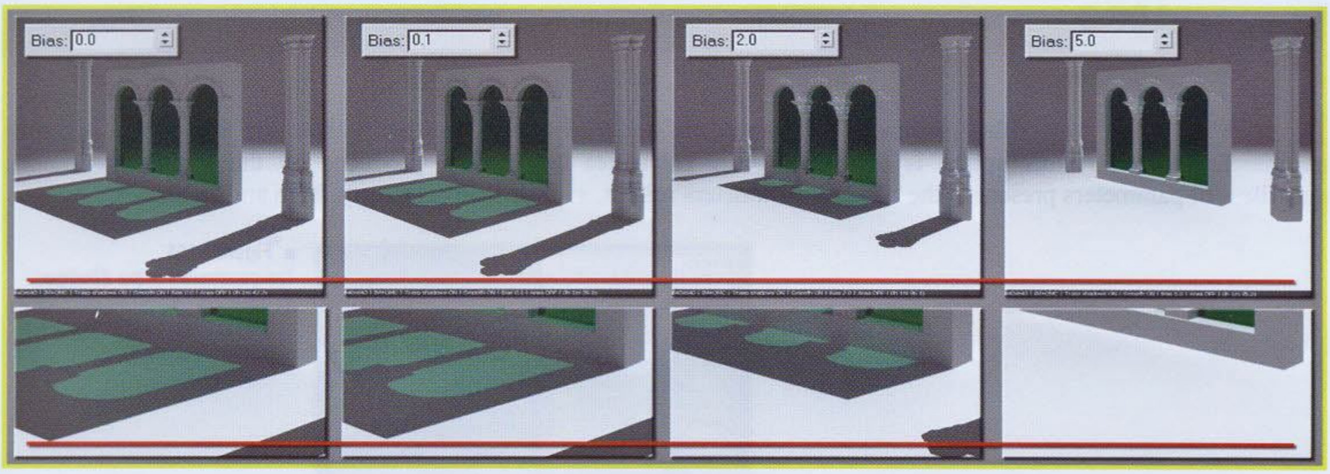
In the following images instead, the ***Transparent shadows*** option has been deactivated. This way it has been possible to use parameters such as Color and Map, located in the Shadow Parameters rollout.



■ **Figure 6.157**
Some parameters from the Shadow Parameters being used.

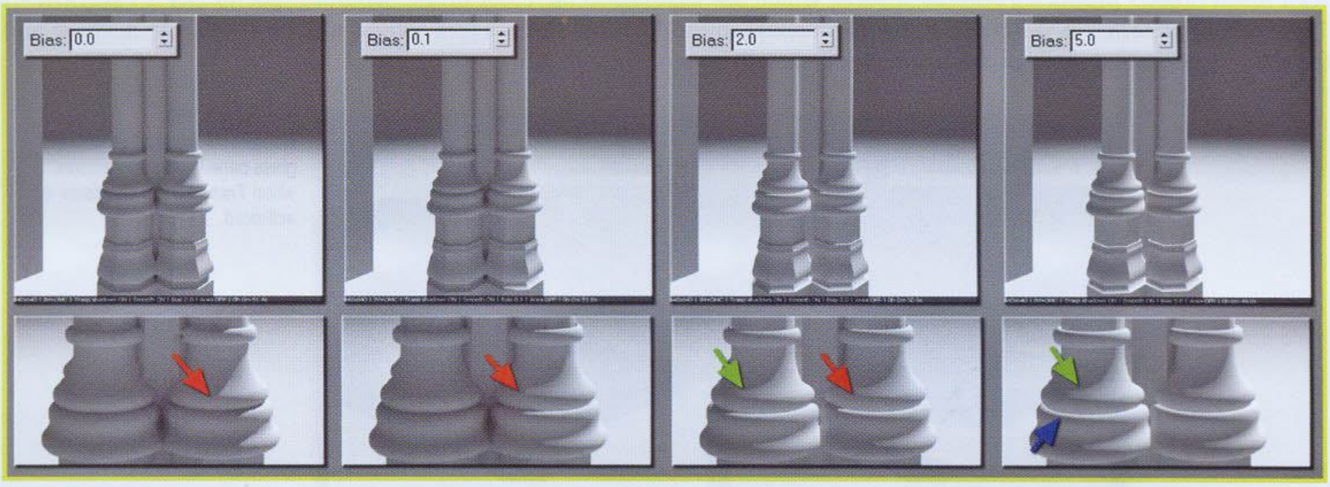
Smooth surfce shadows - if this is active, *VRay* tries to avoid artifacts in the shadows which could occur in the case of geometries with only a few polygons. It is advisable to not activate this parameter with long and thin polygons.

Bias - as an option, according to the user's needs or the situation, *VRay* shadows can be moved in such a way that they near the light source. One may need to use this option if one should have problems like dark stains or self-shadowed surfaces. The default parameter assures a correct calculation of shadows in most cases.



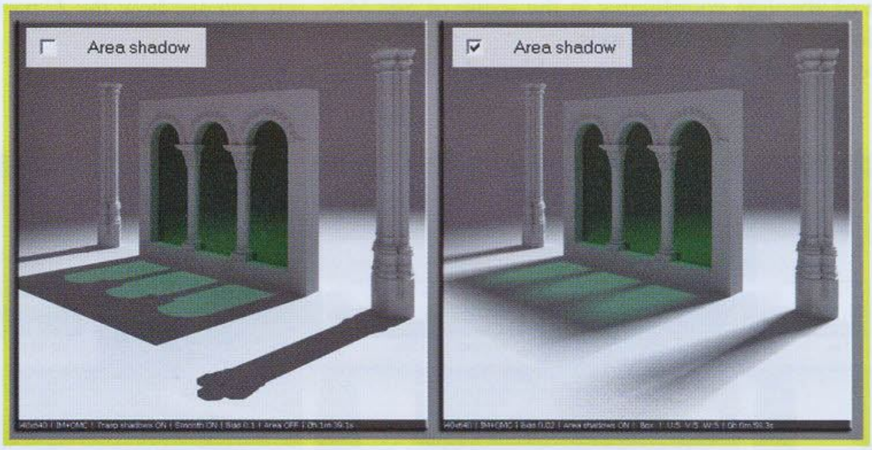
■ **Figure 6.158**
Reduction in the length of shadows caused by increasing the ***Bias*** parameter.

If we take a closer look, we can see that the shadow of the pillar is not correct. The reason is a ***Bias*** setting which is too high.



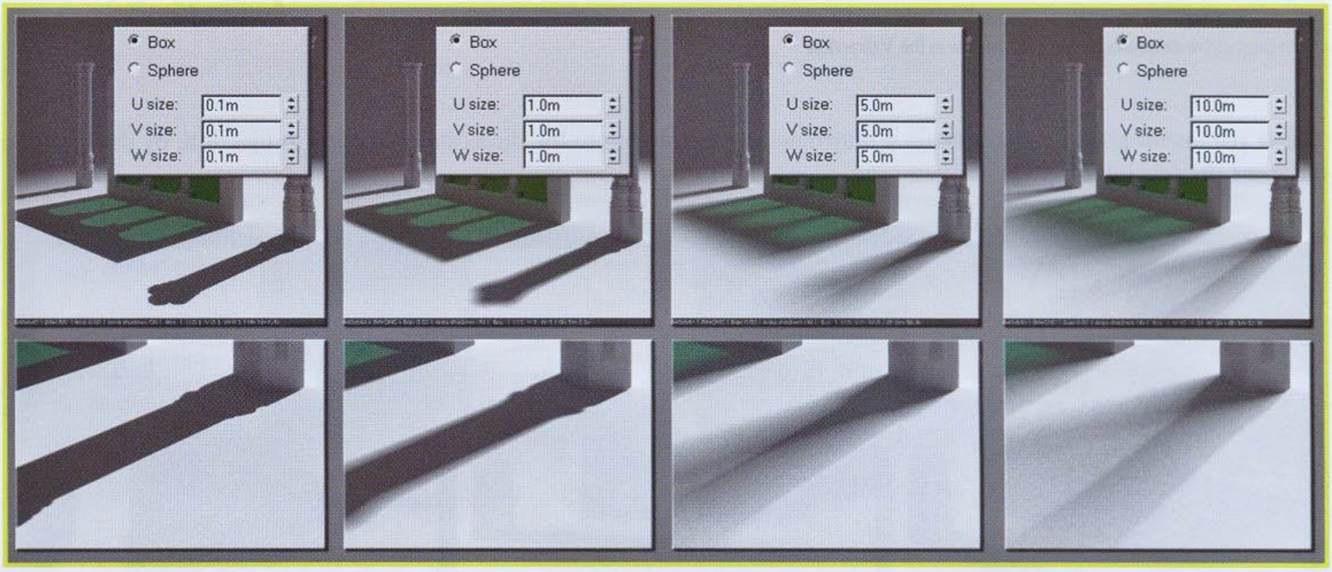
■ **Figure 6.159**
Effect of the ***Bias*** parameter on complex surfaces.

Area shadow - in the examples shown so far, *VRay* has always generated shadows with very defined borders. This situation, in actual fact, does not happen often. This type of shadow could be generated exclusively by a pointlike light source or a very far away one. In nature this is practically impossible, because all light sources have an emitting surface, however small. Also the Sun, which is usually represented with precise borders, does not generate defined shadows. Thanks to the ***Area shadow*** option one can generate shadows with blurry outlines.
Imagine that the Direct Light which has just been created has a perfectly spherical light bulb for its source. In this case, the shadows which are generated will have a definition defined by the incandescent light bulb. Now replace the light bulb with a luminous parallelepiped. The shadow will now have a different appearance because the shape of the emitting geometry has changed. *VRay* allows one to change both the ***U*** radius of the sphere and the ***U***, ***V*** and ***W*** sizes of the ***Box***. These parameters allow one to change the appearance of the area shadows generated by *VRay*, by producing shadows with outlines which are more or less defined.



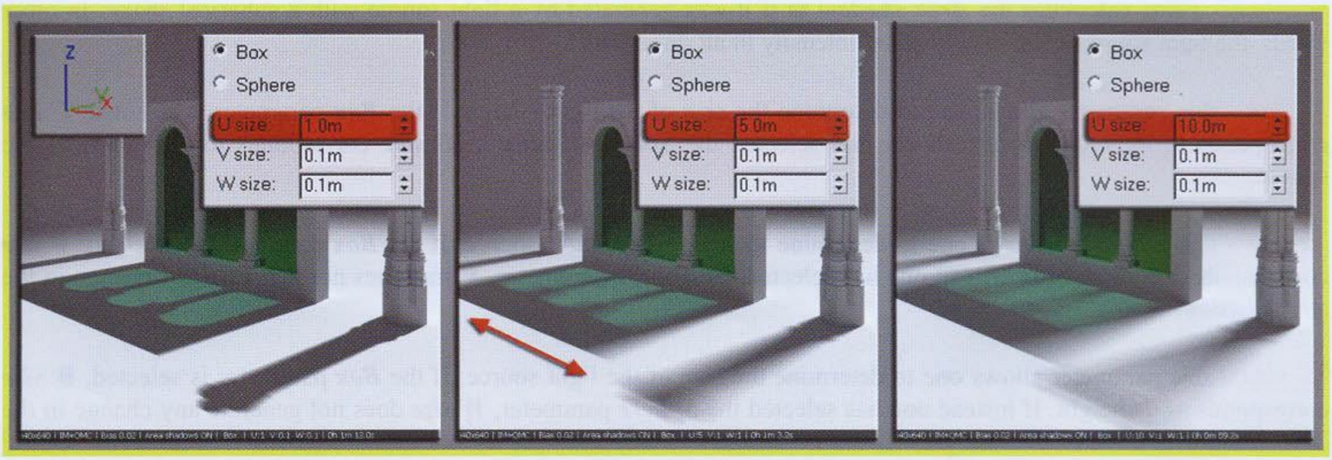
■ **Figure 6.160**
Activation of the ***Area shadow*** parameter. The blurring of the shadow can be seen well.

Box - *VRay* calculates the ***Area shadow*** as if it was generated by a cube-shaped light source.



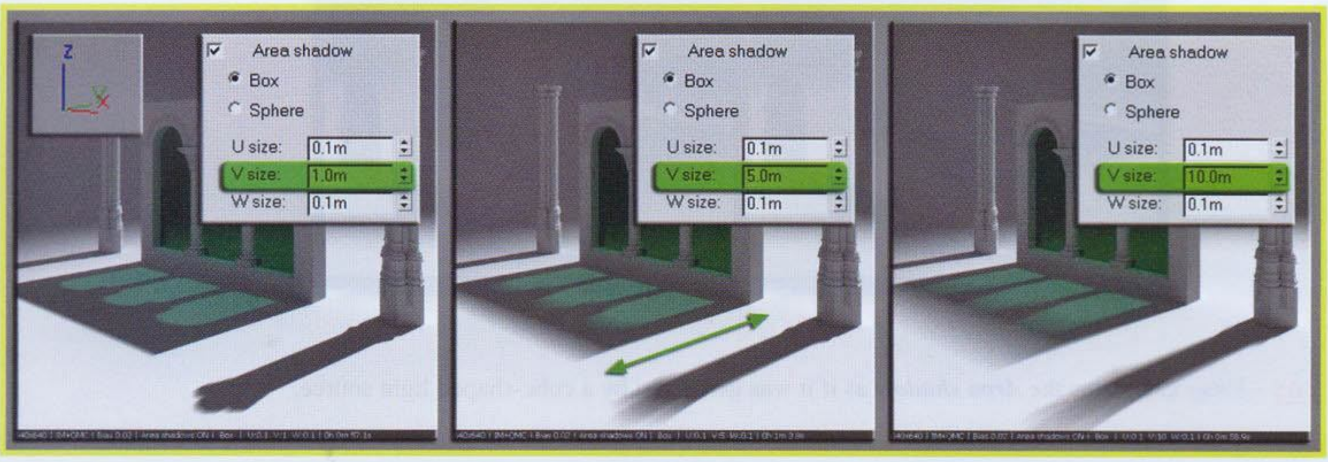
■ **Figure 6.161**
Variation of the size of the **Box** shadow in a constant way in all directions: ***U***, ***V*** and ***W***.

In the previous example, the size of the ***Area shadow*** is the same for all three dimensions and produces a blurred shadow in an equal way for each side.
By changing only the ***U*** parameter and using very low values for the remaining sides, one can witness a different behavior of the shadow.



■ **Figure 6.162**
Variation of the dimension of the box shadow in the ***U*** direction.

Bearing in mind the system of absolute coordinates in the scene of 3ds Max, one can see how the ***U*** parameter allows the forming of blurred shadows only along the X axis.
Instead by changing only the ***V*** parameter, the shadow is blurred only along the Y axis.



■ **Figure 6.163**
Variation of the dimension of the box shadow in the ***V*** direction.

The ***W*** parameter allows one to obtain a blurriness corresponding to the direction of shadow projection, by determining the ***Area shadow*** intensity.

■ **Figure 6.164**
Variation of the dimension of the ***Box*** shadow in the ***W*** direction.

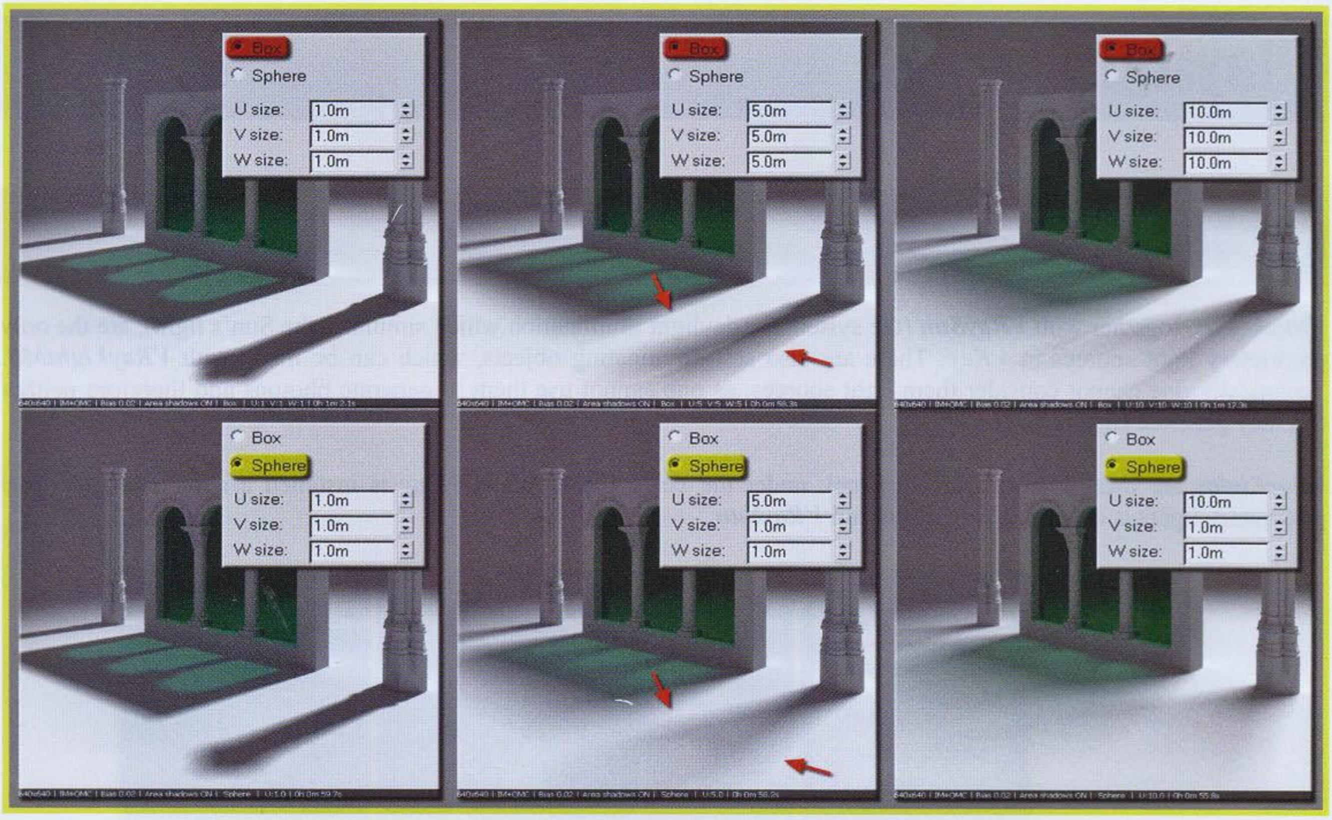
By changing the position of the light source, the blurring of the shadow follows the direction of the Direct Light.

Sphere - *VRay* calculates the ***Area shadow*** as if it was generated by a light source with a spherical shape. In other words, the light source emits rays of equal intensity in all directions.

U size - this parameter allows one to determine the size of the light source. If the ***Box*** parameter is selected, ***U size*** corresponds to its width. If instead one has selected the ***Sphere*** parameter, ***U side*** corresponds to the radius of the sphere.

V size - this parameter allows one to determine the size of the light source. If the ***Box*** parameter is selected, ***V size*** corresponds to its length. If instead one has selected the ***Sphere*** parameter, ***V size*** does not generate any change in the ***Area shadow***.

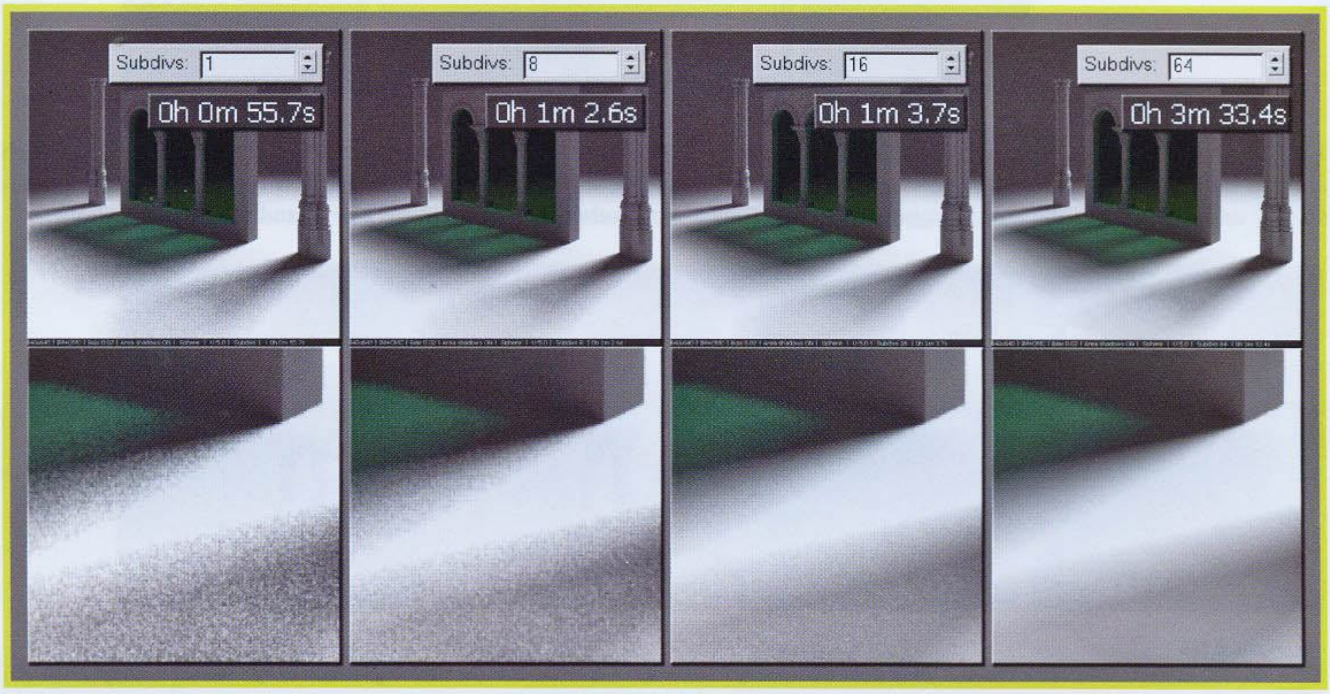
W size - this parameter allows one to determine the size of the light source. If the ***Box*** parameter is selected, ***W size*** corresponds to its height. If instead one has selected the ***Sphere*** parameter, ***W size*** does not generate any change in the ***Area shadow***.



■ **Figure 6.165**
Comparison between the ***Box*** and ***Sphere*** modes for the calculation of the ***Area shadow***.

If we observe the previous image, at equal ***Area shadow*** size, using the ***Sphere*** mode produces a more uniform shadow along the edges. As emphasized by the arrows, the side borders of the shadows generated in ***Box*** mode are much more defined compared to when using ***Sphere*** mode. This last mode, at equal size, produces a more blurry shadow.

Subdivs - in zones of transition between the part of surface which is in the shadows and the part which is illuminated, one can see a certain amount of noise. In order to increase the quality of shadows and therefore to reduce noise, it is necessary to increase the ***Subdivs*** parameter. Thanks to this value it is possible to control the number of samples which *VRay* uses to compute the ***Area shadow***. A greater number of subdivisions means a better level of quality and higher calculation times.



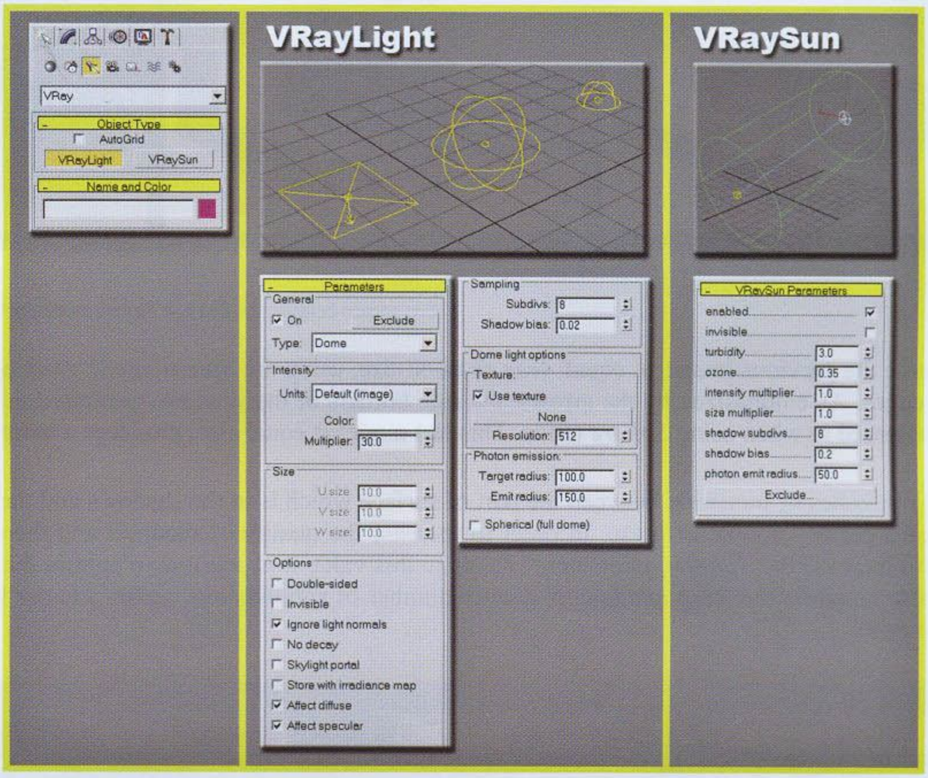
■ **Figure 6.166**
Increase in quality of the ***Area shadow*** due to the ***Subdivs*** parameter.

VRayLight

INTRODUCTION

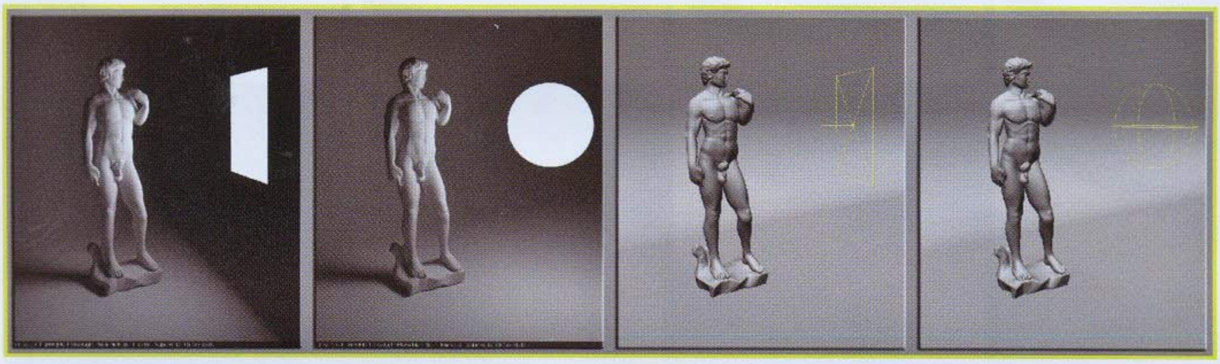
VRayLights, together with ***VRaySun*** (the system for daylight illumination which simulates the Sun's light), are the only proprietary light sources in *VRay*. There are also self-illuminating objects, which can be made with ***VRayLightMtl***. Technically one cannot consider them light sources, as one cannot use them to generate photons and therefore neither can they generate caustic effects.

VRayLights are located in the Create panel, under the voice Light. When *VRay* is installed a new series of *VRay* proprietary lights is created: ***VRayLights*** and ***VRaySun***.



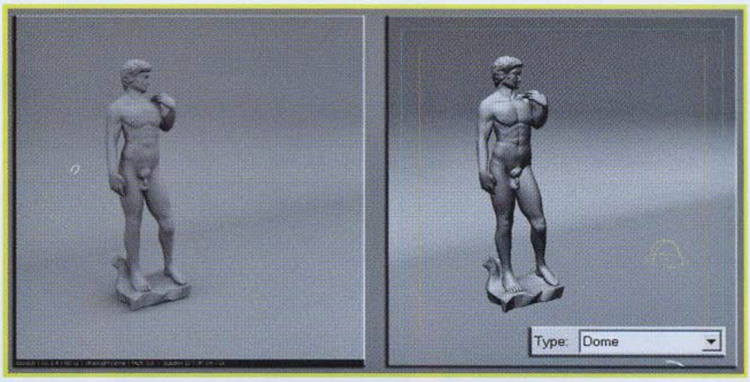
■ **Figure 6.167**
Two light VRay light sources with default parameters.

VRayLights, unlike 3ds Max lights, are always and only ***Area*** lights. They are light sources which have a surface which emits a luminous flux. They can be very small, but always have a certain area. There are two types of sources: plane-shaped or spherical. As the name suggests, the first type is none other than a luminous plane, and the latter a sphere.



■ **Figure 6.168**
The two main types of illumination available in VRay: a plane-shaped light source and a spherical one. The shadow generated by light sources is always of the ***Area*** type.

There is also a third type of light called ***Dome Light***, which like the others is also contained within ***VRayLight***. It can be selected with a pull-down menu, but should be considered more like a system for generating global light diffusion without using the GI, more than an actual light source. For anyone with a bit of experience in using 3ds Max, ***Dome light*** is very similar to Skylight in 3ds Max.



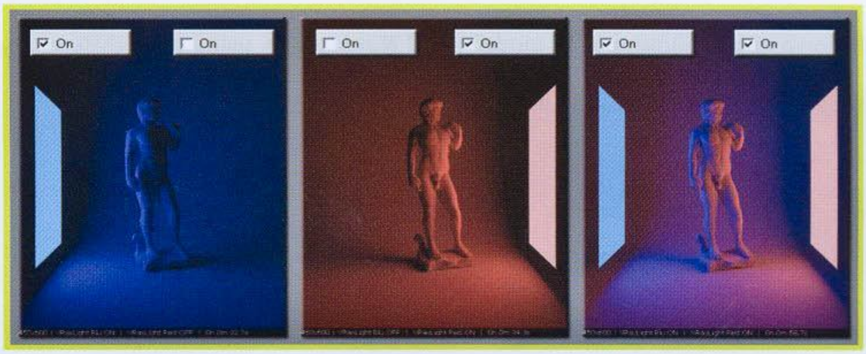
■ **Figure 6.169**
Dome Light. The left-hand image has been created without the use of Global Illumination.

The remaining parameters allow one to determine the intensity of the luminous flux, the size of the light source, the color and other properties such as visibility during rendering, the effect on reflective objects an so on.

PARAMETERS

General

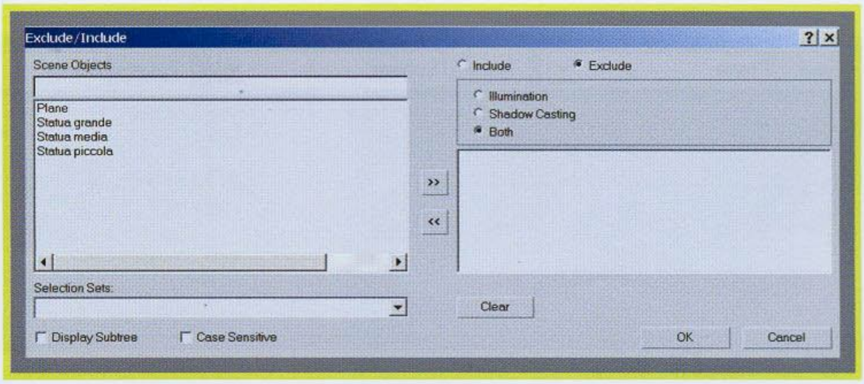
On - this activates or deactivates the light source.



■ **Figure 6.170**
Activation or deactivation of ***VRayLights***.

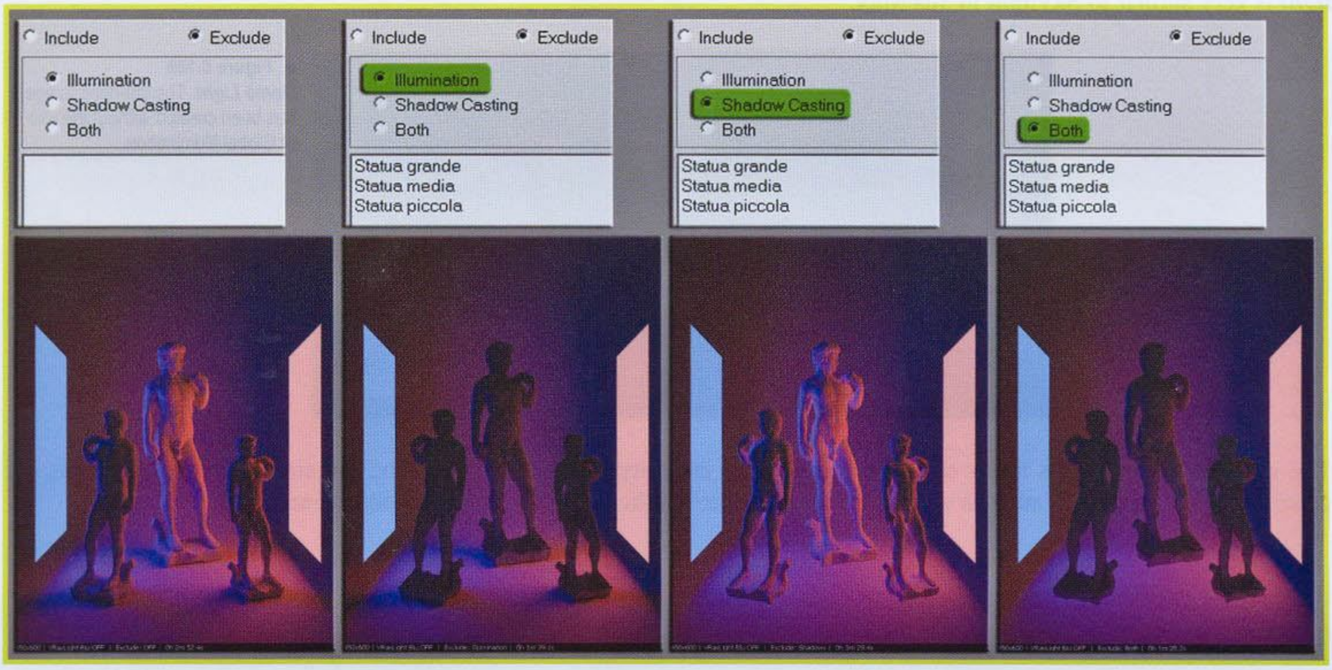
In this scene two identical (except for their color) plane ***VRayLights*** have been used. By deactivating them, their presence and visibility is none, as if they were not in the scene.

Exclude - by clicking on this button, a screen appears with which it is possible to define exactly which objects should be illuminated, which can project shadows and which can do both. This is all referred to the selected ***VRayLight***.



■ **Figure 6.171**
Chart for illumination parameter management and shadow projection.

To the left of the chart the names of the geometries present in the scene are to be found. By moving some of these meshes right, one can decide if they should be included or excluded by illumination, of they should project shadows, or both of these actions. In order to quicken this process, one can use Selection Sets.

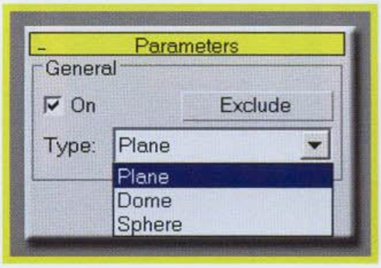


■ **Figure 6.172**
The options in the ***Exclude/Include*** chart being used for both ***VRayLights***. In this example, ***Exclude*** has been used.

In the render on the far left, one can see how the sculptures are perfectly illuminated and how shadows are correctly represented. In the second render, although shadow are perfect, the statues are not affected by the direct illumination of the *VRayLights*. They are however illuminated indirectly. In the third render shadows are not present, and in the last one, they are not illuminated and neither do they project shadows.

Type - *VRay*, with this pull-down menu, allows one to modify the ***VRayLight*** by using three different types of source.

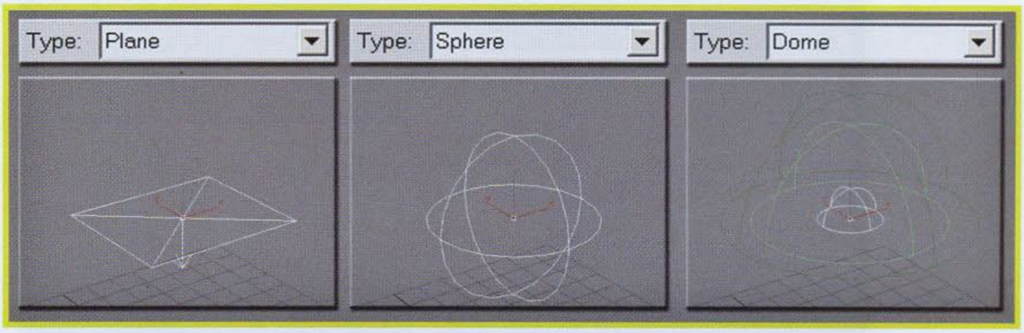
■ **Figure 6.173**
The three types of light within the ***VRayLight***.



Plane - the ***VRayLight*** is made via a 2D plane, which emits light from the entire plane surface.
Sphere - the ***VRayLight*** is represented by a 3D plane, which emits light from the entire spherical surface.
Dome - this type of ***VRayLight*** allows one to simulate a diffuse light typical of the GI, but without actually using Global Illumination. The result is similar to what one obtains by using the Skylight object in 3ds Max.

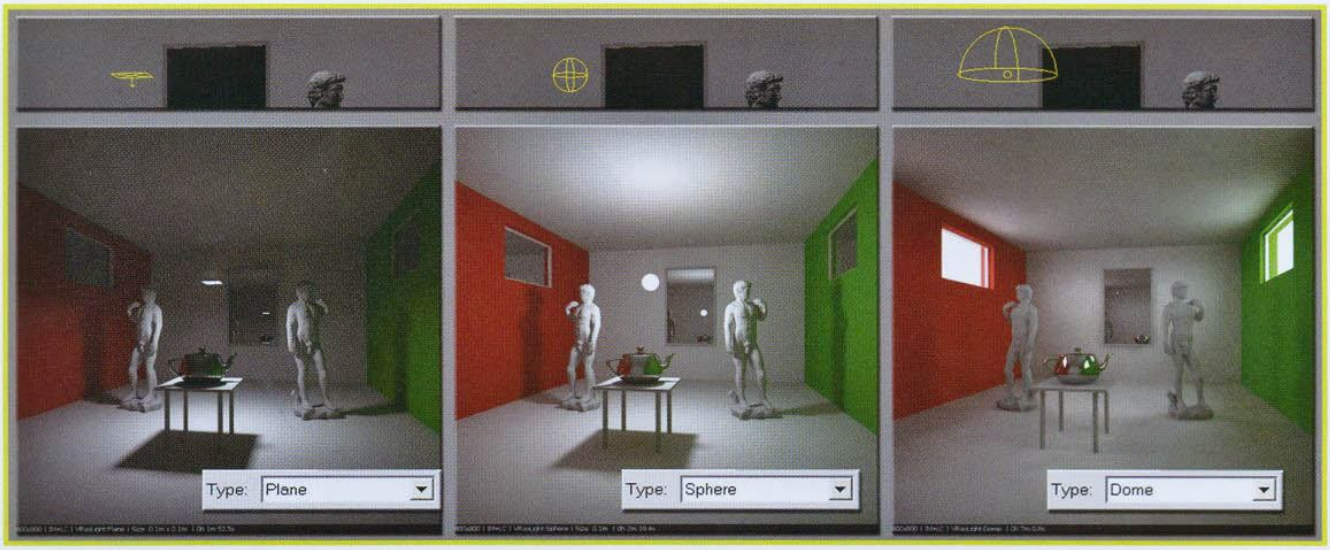
By changing the ***Type*** parameter, the representative icon of ***VRayLight*** in the viewport is adapted to the new type.

■ **Figure 6.174**
The three different icons of VRay lights.



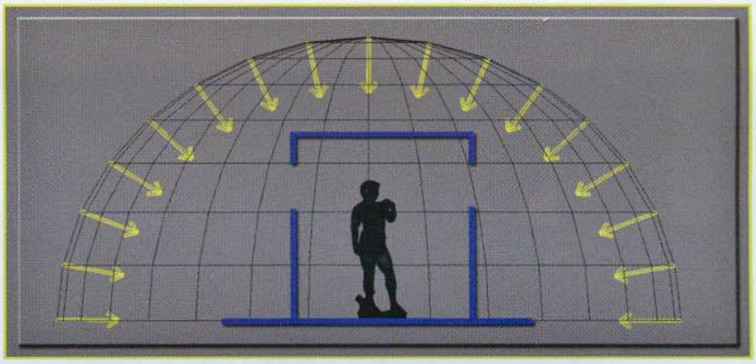
VRAYLIGHT DOME

The first two types, ***Plane*** and ***Sphere***, represent exactly what the size of the panel or the luminous sphere which emits light will be. Instead Dome is an icon which identifies the type of light and its direction, but has nothing to do with what the real result will be. In order to understand this particular type of light better, observe the following image.



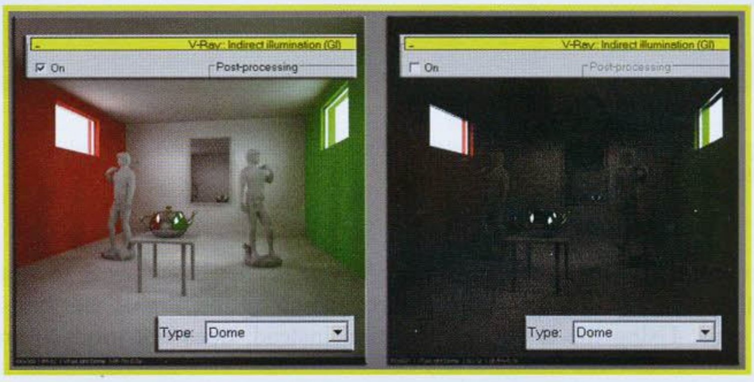
■ **Figure 6.175**
The three types of lights in action, used with the Global Illumination not active.

The ***Plane*** type light emits a luminous flux from the plane geometry following the direction of the arrow in the icon, as if it was a light source, whereas ***Sphere*** emits light in every direction, as if it were a light bulb. The ***Dome Light*** is a bit different. Its use is not advisable for indoor renders, because its function is to create a spherical cap which emits infinite rays inwardly.



■ **Figure 6.176**
Scheme showing the way ***Dome Light*** functions.

The ***Dome Light*** can be represented simply as a semisphere containing the whole scene, a bit like *VRay*'s ***skylight***. Unlike ***skylight***, however, which requires the GI to be activated, this is not necessary with ***Dome Light***, because the generation of light is an intrinsic quality of the ***Dome Light*** itself, also without Global Illumination.

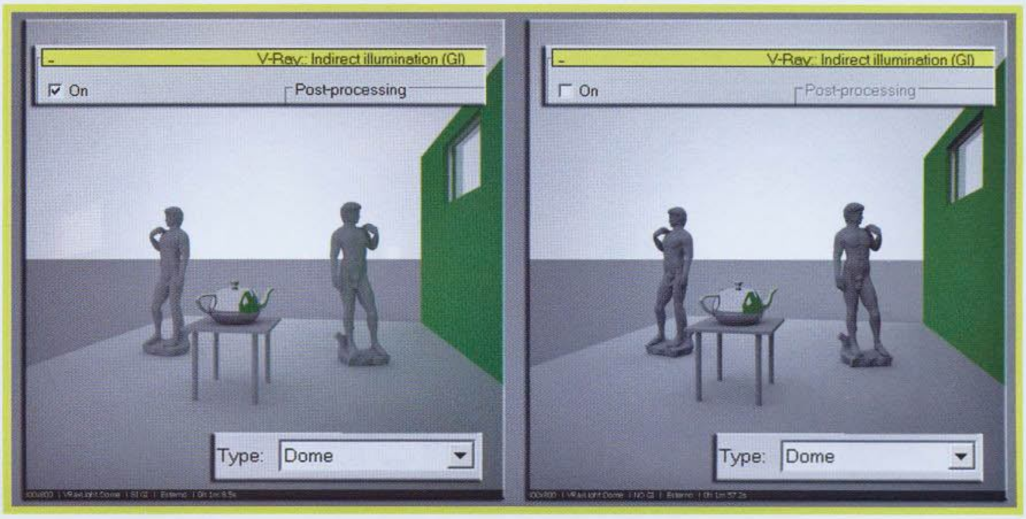


■ **Figure 6.177**
Calculation of an internal scene with activation and deactivation of the GI with a ***Dome VRayLight***. For internal renders with GI active, it is preferable to use VRay ***skylight*** instead of ***Dome Light***.

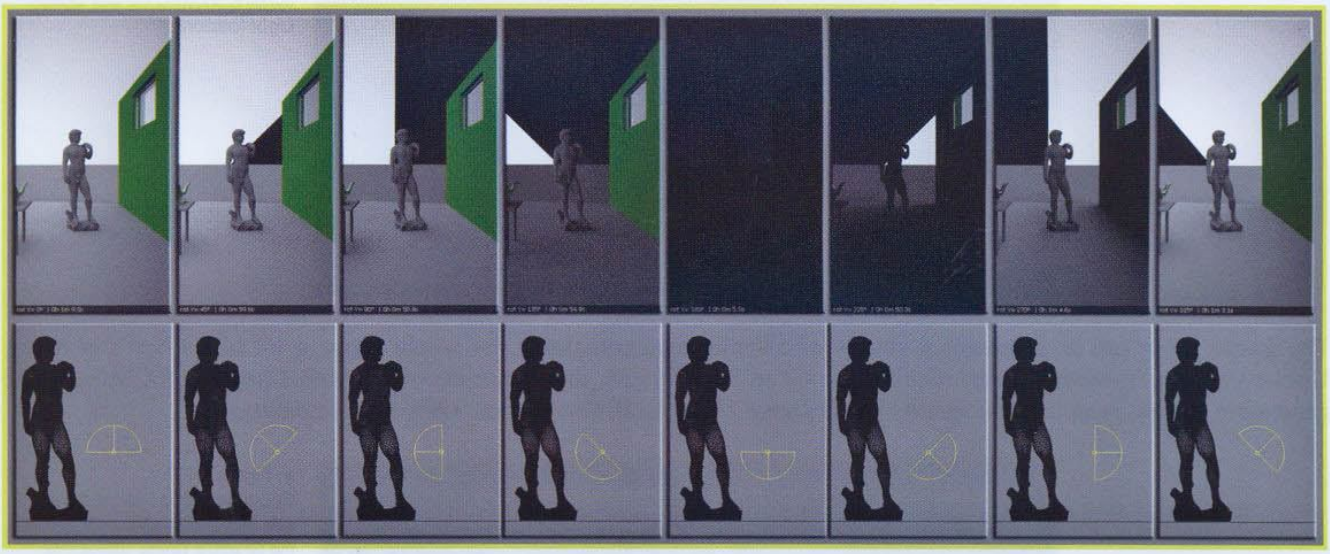
Definitely in internal scenes without GI, the illumination generated by ***Dome Light*** is poor and hardly realistic because its light is supposed to bounce on objects and spread in a uniform way. Without GI the light in the scene is only direct and therefore does not have secondary bounces.

This "flaw" of ***Dome Light*** is not as important in outdoor scenes, where the result with or without GI is not all that different.

■ **Figure 6.178**
Outdoor use of ***Dome Light***, with GI active and not.



Having very few obstacles, ***Dome Light*** can efficiently illuminate the scene without that many problems. There are no effects of color bleeding or other particular problems due to the use of the GI, but with this method, for instance, we can be sure to avoid flickering problems during animations and it is possible to setup a scene quickly without having to activate the GI.
There is one consideration which must be made regarding ***Dome Light***: rotating it changes the illumination of the scene. Observe the following examples.

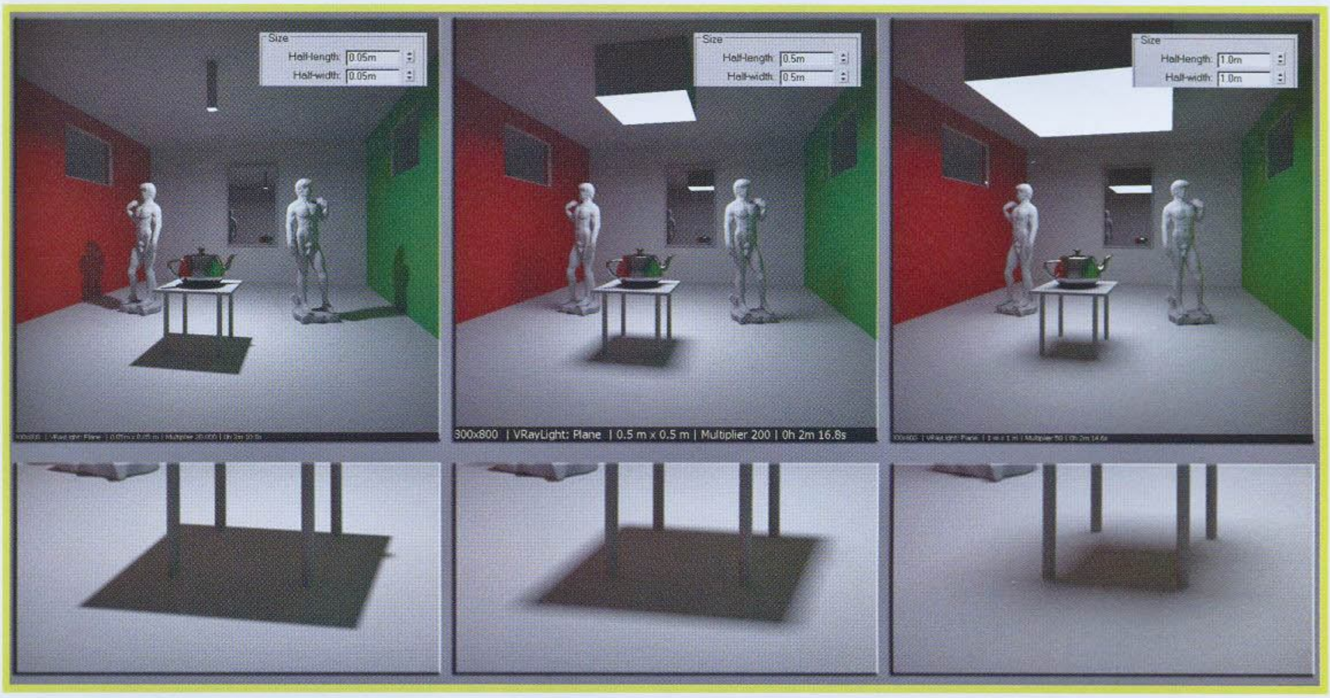


■ **Figure 6.179**
The effects of rotating the ***Dome Light***. Notice how, by applying a 270° rotation without the GI, the vertical wall is completely dark and lacking illumination. With a 180° rotation the scene is completely dark.

The effects of ***Dome Light*** rotation are obvious: general illumination is coordinated with the rotation of ***Dome Light***, as is the Background also. In this case, ***Dome Light*** has been rotated along the Y axis, but the same results would have been obtained along the X axis. Using the Z axis for rotation instead, does not produce any change, like also the translation of the ***Dome***.

VRAYLIGHT PLANE

The *Plane* type *VRayLight* is very easy to understand. It is simply a luminous plane. Whereas for *Dome Light*, the main parameters are, as well as intensity and color, its direction, for *VRayLight Plane* the main parameters are size adjustments, which allows one to influence shadows as well. In general, the smaller *VRayLight* is, the sharper shadows appear. Vice versa, otherwise shadows are smoother and gentler.



■ **Figure 6.180**
Variation in the size of the *VRayLight Plane* influences the type of shadow directly.

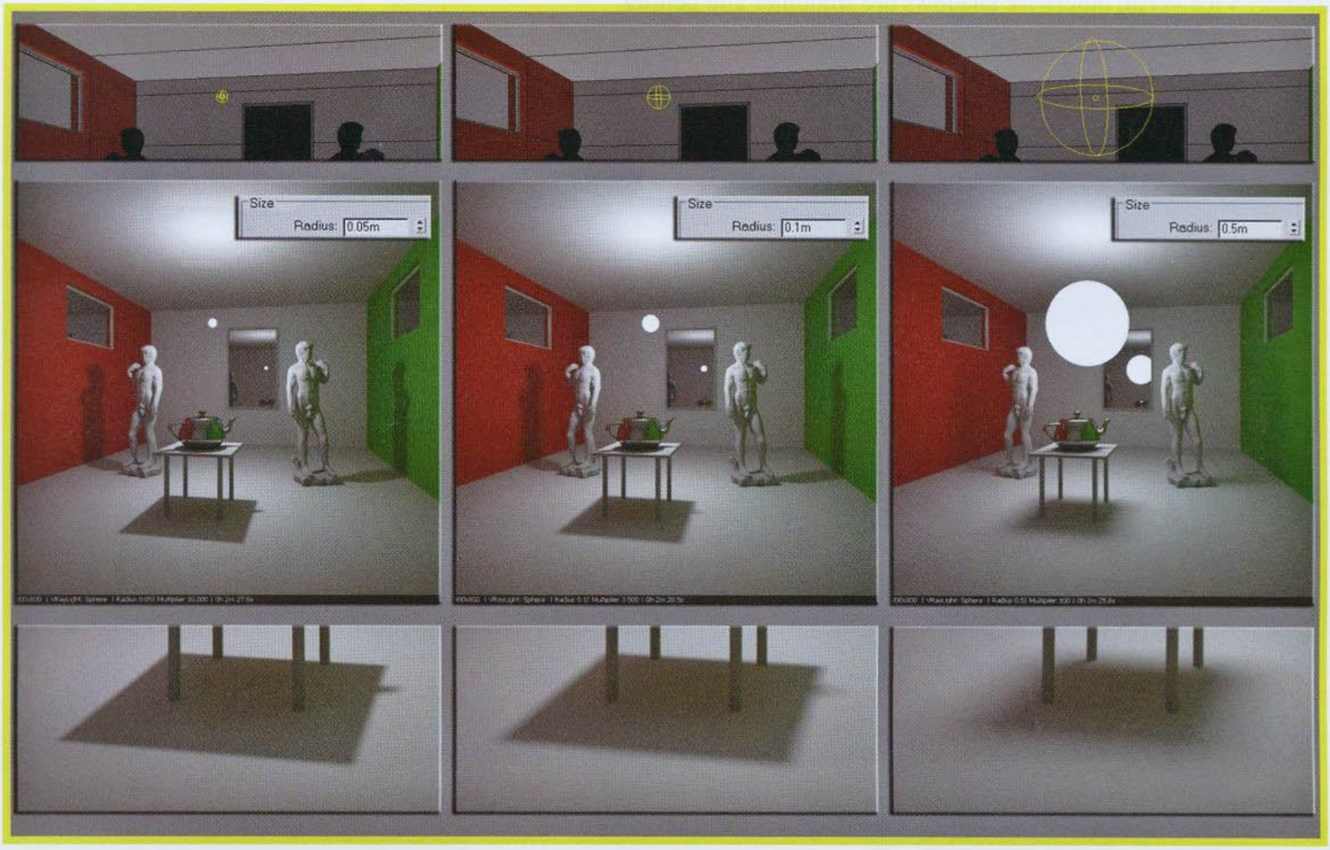
The *VRayLight Plane* is flat. Its rotation position is of fundamental importance for illumination.

■ **Figure 6.181**
Change in illumination due to rotation of the *VRayLight Plane*.

There are many other options which concern *VRayLights*, and according to the type selected, they can produce different results. Later on, we will take a look at all the other parameters and try to analyze their behavior. For now we will focus our attention on the two types, *Plane* and *Sphere*, and try to understand their basic behavior.

VRAYLIGHT SPHERE

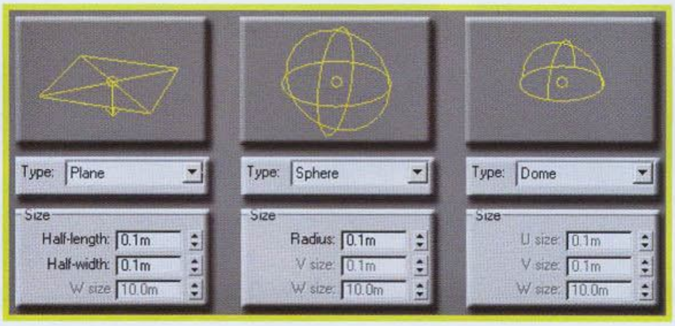
This type of light is the simplest, as a concept. It is a completely luminous sphere which emits light in all directions in a uniform way from every point on its surface. Similarly to the ***VRayLight Plane***, also for the spherical type, the size determines the smoothness of shadows. The greater the radius of the sphere, the smoother the shadow becomes.



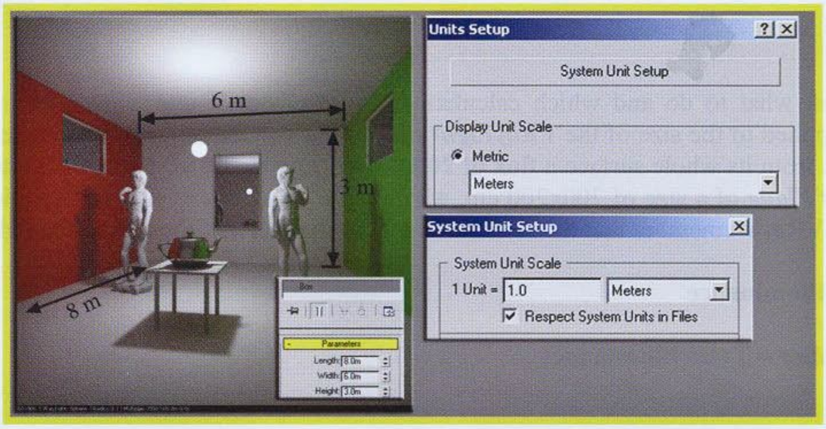
■ **Figure 6.182**
Variation of shadows according to the size of the sphere-type ***VRayLight***.

Size - through this section it is possible to specify the size of ***VRayLights***. In ***VRay*** there are two luminous sources: plane and sphere types. According to the type of light, one is shown, respectively, half the width and half the length of ***VRayLight Plane*** or the radius of the ***VRayLight Sphere***. The ***Dome Light*** does not have size adjustments.

■ **Figure 6.183**
The parameters for re-sizing ***VRayLights***.



Before describing the parameters it is necessary to be well aware of what unit of measurement ***VRay*** uses for its calculations. ***VRay*** works internally by using generic units. More precisely, ***VRayLights*** assume that 1 Generic unit corresponds to 1 meter. For this reason, the scene used for the test measures 8x6x3 generic units, which correspond to 8x6x3 m. Soon we will show how the units of measurement of 3ds Max interact with the intensity values of ***VRayLights***. For know it is enough to know that the units of measurement used in the scene are meters.

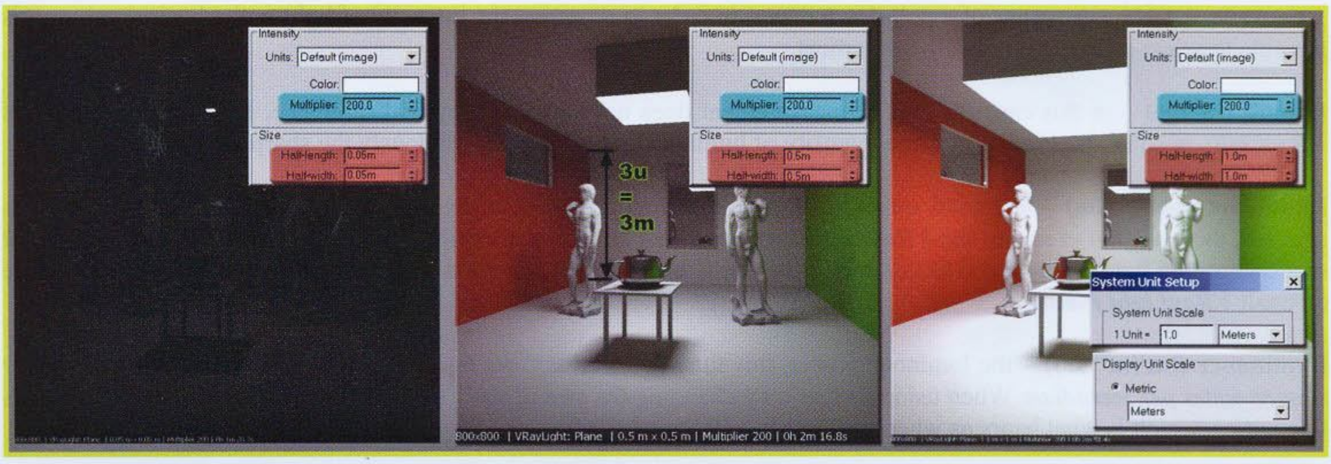


■ **Figure 6.184**
Unit setup of the scene and reference size of the models.

In the following images the variation of size of the ***VRayLight*** is always accompanied by a variation of the intensity of light itself. This way the room is illuminated always in the same way. This happens because the ***Default image*** unit of measurement has been used. By using this value, an increase or decrease in ***VRayLight*** size brings an increase or decrease in luminosity of the light itself, as well as a variation of shadows. A 2D plane, which emits light from the entire plane surface.

■ **Figure 6.185**
Variation of ***VRayLight Plane*** size. The shadow produced by a ***Plane*** of large size is smoother compared to a mall-size one. The ***Multiplier*** has been modified in accordance with the size of the light source in order to obtain the same amount of intensity.

The ***Size*** of the ***VRayLight Plane*** represents half the length and width of the light source. This means that in order to obtain the exact size one must multiply the ***Half-length*** and ***Half-width*** values by 2. Notice how the ***Multiplier*** value, which is the parameter which allows one to define the quantity of light emitted by ***VRayLight***, decreases with the increase in light surface. If this were not the case, by using the same ***Multiplier*** value all the time, decreasing the area of emission of luminous flux, also the luminosity of the scene would decrease.
Otherwise the scene would be overexposed.



■ **Figure 6.186**
Variation in the size of the ***VRayLight Plane*** and the effect on the luminous flux.

Intensity

At this stage one might ask what to do and which calculations to carry out if one wanted to maintain the same luminosity if changes are applied to the size of the ***VRayLight Plane***. The calculation is quite simple. Imagine a plane of 100x100 cm which emits from its whole surface a flux of 200. If one doubles the size and wishes to obtain the same intensity from a ***VRayLight Plane*** of a size of 200x200 cm, which is the ***Multiplier*** value that should be inserted? By using the Size parameter of VRay (remember that the indicated value represents half the size of the light source):

Area x Multiplier = Total Illuminance

(Width x Length) x Multiplier = Total Illuminance

(0.5m x0.5m) x 200 = 50

50 is the total illuminance that ***VRayLight*** emits. All that is left to be done now is to spread this flux on a greater area, dividing the illuminance by the new size of the ***VRayLight***.

Total Illuminance / Area = new Multiplier

50 / (1m x 1m) = 50

If the ***VRayLight***, instead of increasing, decreases in size, for example 1/10 for each side, the result would be:
50 / (0.05m x 0.05m) = 20,000

Exactly like the last example.

As far as ***VRayLight Sphere*** is concerned, one applies the same formulas used for ***VRayLight Plane***. The only thing that changes is the formula for the area which is:

Area = Area Sphere = 4п2

There is another consideration to be made. So far we have used default units of measurement for the ***VRayLight***. In this case, the luminous flux is closely connected to the size of the ***VRayLight***. Later on, when we cover the units of measurement thoroughly, we will see that there are some which do not generate any variation in the emitted light, leaving it constant, regardless of the size of the ***VRayLight*** being changed.

It is therefore understandable how the size of the ***VRayLight***, the ***Multiplier*** and the unit of measurement used are closely connected to each other.

Multiplier - through the use of this parameter one can modify the intensity of the ***VRayLight***. Notice how the value chosen changes according to the unit of measurement. The units can be set via the pull-down menu ***Units***.
By changing the unit of measurement, it is ***VRay*** which takes care of conversion, for example between ***Luminous power*** and ***Radiant power*** or ***Luminance*** and ***Radiance***, by inserting the correct ***Multiplier*** value.

Intensity Units - by means of a pull-down menu, in order to represent the intensity of light with physical values it is possible to select different units of measurement.

Default (image) – in this case the unit of measurement does not have any physical reference. By varying the size of ***VRayLight***, at equal ***Multiplier***, also the intensity of luminous flux varies and obviously the sharpness of shadows.

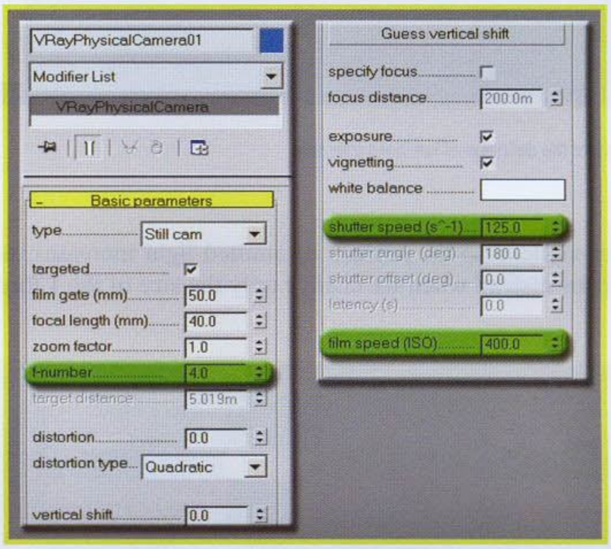
Luminous power (lm) – the total quantity of visible luminous power is expressed in ***lumen***. When this mode is selected, the variation in size of the ***VRayLight*** does not affect luminous intensity. The amount of light and sharpness of shadows are no longer connected to this factor. It is possible to obtain shadows with defined outlines both with a large or small ***VRayLight Plane***.

Luminance (lm/(m^2*sr)) – the luminous power, this time not total but per unit of surface, is expressed in ***lumen per square meter per steradian***. When using this unit, by changing the size of the ***VRayLight***, also the intensity of luminous flux varies, which is what happens when one uses the ***Default*** unit too.

Radiant power (W) - the total quantity of visible luminous power is expressed in ***Watts***. When this mode is selected, the change in size of the ***VRayLight*** does not affect luminous intensity. The same phenomenon can be witnessed when using the ***Luminous power*** quantity unit. Remember that an incandescent light bulb of 100W converts only 2-3% of absorbed energy into light.

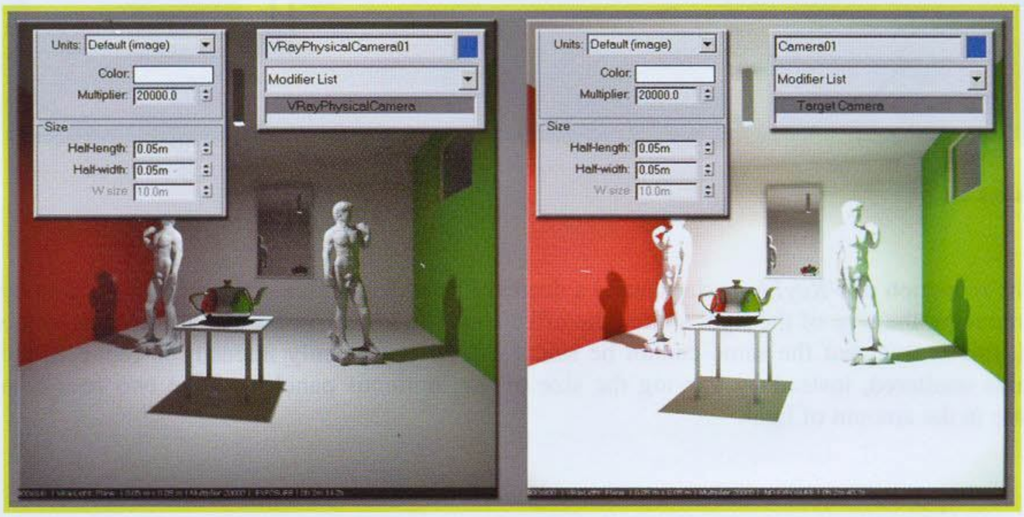
Radiance (W/(m^2*sr)) - luminous power, not the total amount but per unit of surface, is expressed in ***Watts per square meter per steradian.*** By using this unit, if one changes the size of the ***VRayLight***, also the intensity of the luminous flux changes, as it does when one uses the ***Default*** and ***Luminance*** units.

Before getting into explanations, it is best to make one point clear. *VRay* equals 1 unit of 3ds Max to 1 meter. Knowing which unit of measurement one is using is fundamental for a successful render, especially in this case, where one is assigning real parameters to the ***VRayLights***. This is not enough however. It is necessary for the observation point to also be physically correct. Because of this, in order to use ***VRayLights*** efficiently with real units, one must make use of ***VRayPhysicalCamera***. Otherwise, the units of measurement of ***VRayLights*** would no longer make any sense and the result would not be correct. In fact the more practical and attentive users of *VRay* may have noticed that for lights so far we have used multiplier values which in the traditional use of *VRay* without ***VRayPhysicalCamera*** would have proven to be too high. This is because so far we have always used a ***VRayPhysicalCamera*** with the following values.



■ Figure 6.187
The parameters of the ***VRayPhysicalCamera*** used in the tests on ***VRayLights***.

If 3ds Max cameras or ***VRayPhysicalCameras*** are used with the exposure parameter disabled in combination with real physical units in ***VRayLights***, the render would end up being overexposed.

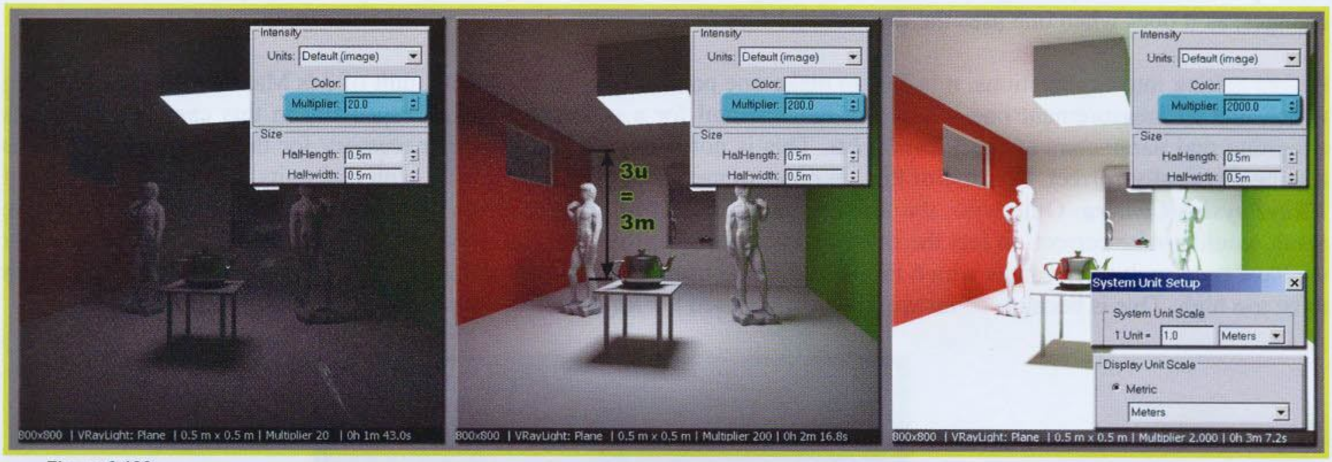


■ Figure 6.188
Overexposed render, due to the use of a 3ds Max Camera.

It could be that the concept of ***Multiplier*** is still not quite clear, as it has been used so far only with the default unit of measurement. When we pass on to the use of real units, such as ***lumen*** or ***Watts***, everything will become clearer and more understandable.

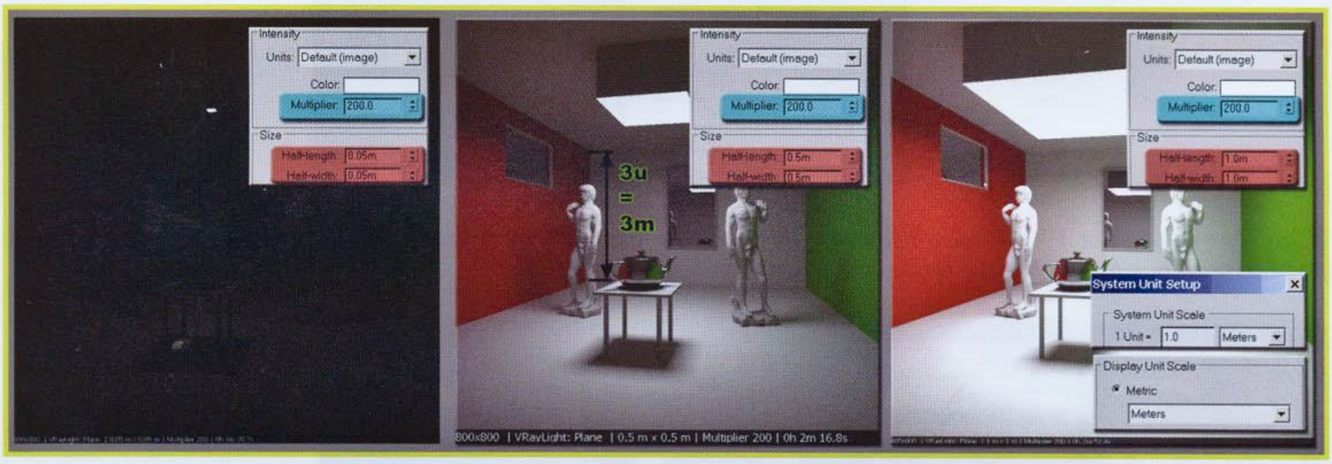
DEFAULT: (image)

The ***Default (image)*** unit is the first one that was made available in ***VRay***, and for a long time was the only one, until physical units were introduced with version 1.5. As the name suggests, generic measurement units are used.



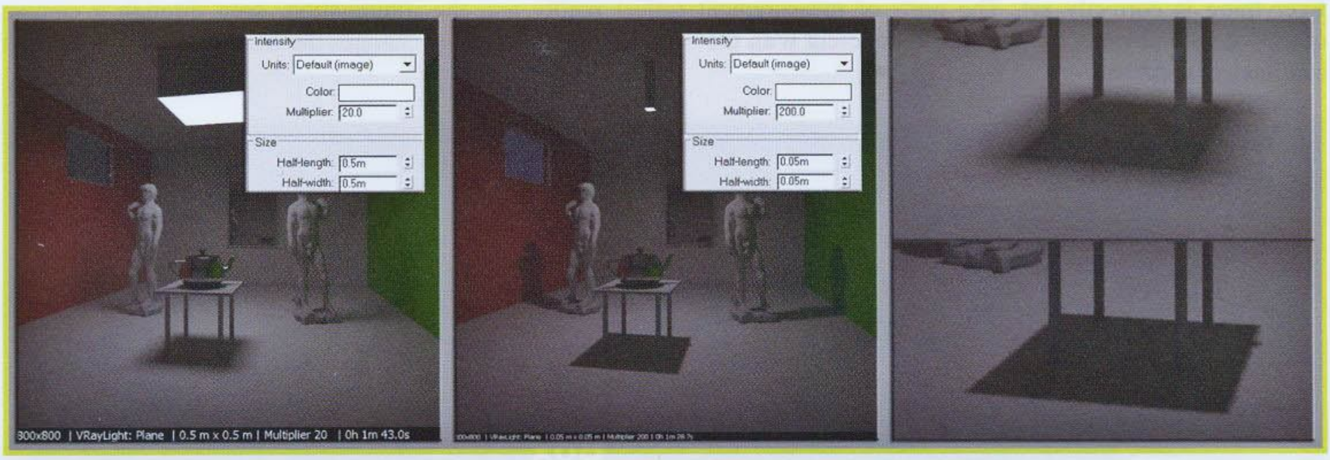
■ **Figure 6.189**
Example of ***Default*** units being used for the definition of luminous intensity.

By increasing the ***Multiplier*** value, also the amount of emitted light increases, which is quite an obvious observation. Now we shall see what happens instead when we change only the size of the ***VRayLight*** with the ***Default*** unit.



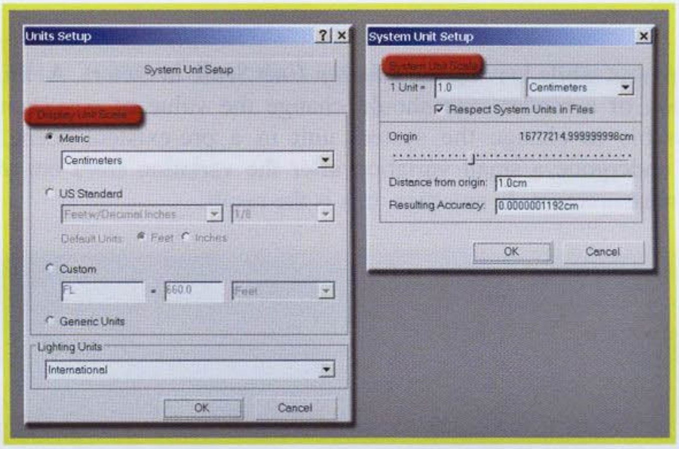
■ **Figure 6.190**
Different ***VRayLight*** size with the ***Default*** unit.

In this particular situation, the reduction of ***VRayLight*** size causes a decrease in luminous flux. One might think that by lowering the ***Multiplier*** or reducing the size of the light source the effects are the same. As far as the intensity of the luminous flux is concerned, this is true, but the same cannot be said for shadows. If only intensity is reduced, the sharpness of shadows remains unaltered, instead by varying the size of the luminous panel we have two results: a sharper shadow and a decrease in the amount of light.



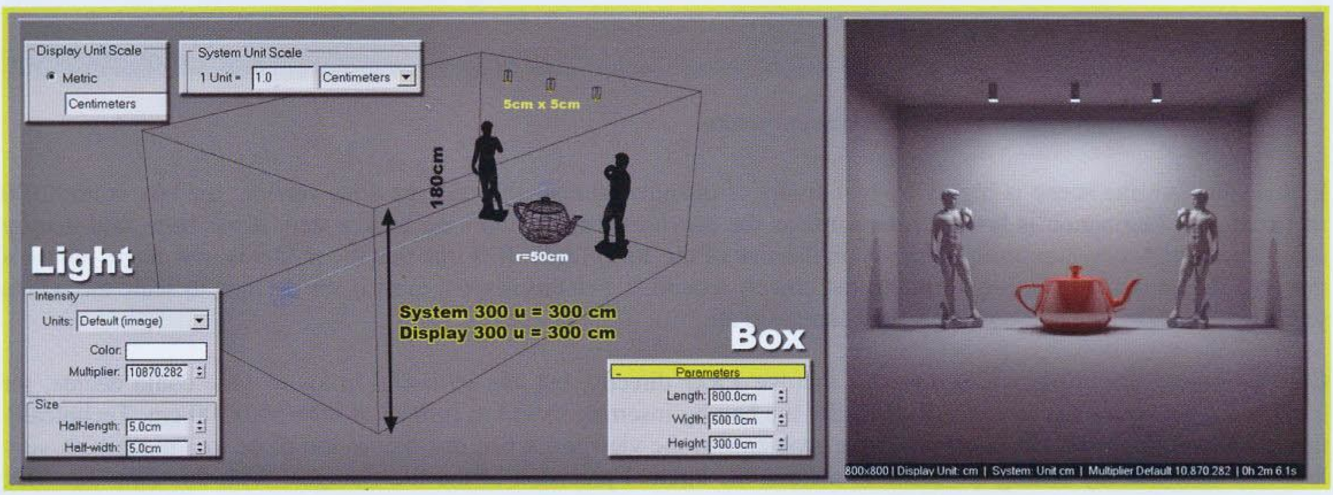
■ **Figure 6.191**
Two similar results obtained in different ways. In the first case it the choice has been made of decreasing light intensity compared to the initial values. This way the smoothness of shadows has been preserved. In the second case only the ***VRayLight*** size has been reduced. This way luminosity has been decreased but shadows are sharper.

At this stage, we shall see what happens by changing the scale of the model. The first thing to be done, when one creates a file in 3ds Max, is to set the unit of measurement of the file. 3ds Max allows for two types of settings. The first one only regards only the units with which parameters are shown. The second one allows one to modify the system unit for files.



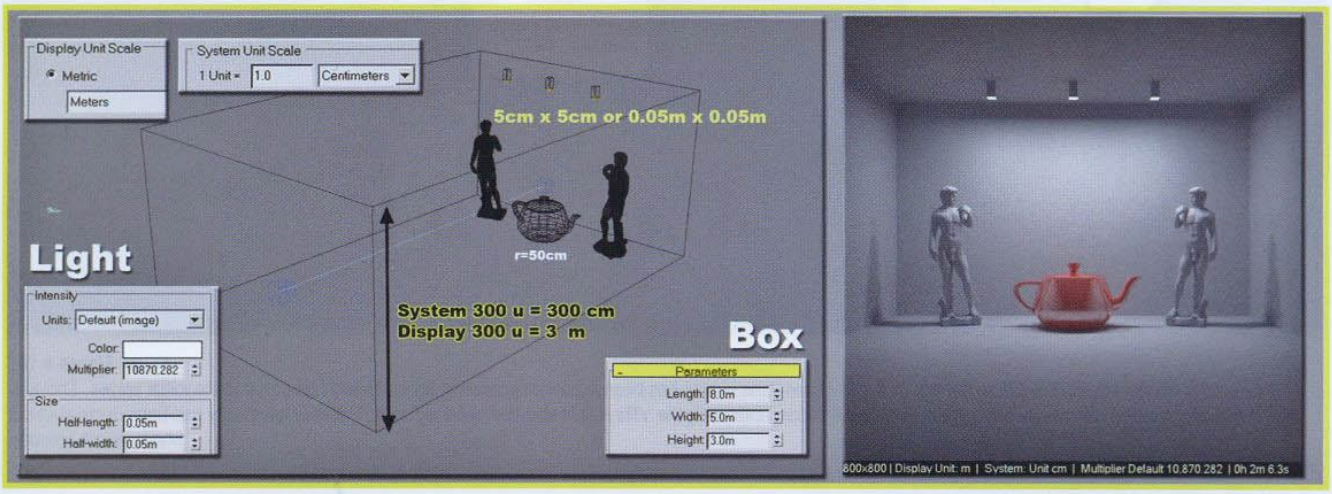
■ **Figure 6.192**
The two windows with which one can set the measurement units of 3ds Max.

In order to understand the use of these windows better, we will give an example. Firstly one creates a new file and then the two units of measurement must be set to cm. Then one creates a room, with a size of 800x500x300 cm. Inside one places a teapot with a radius of 50 cm, two man-size statues and three ***VRayLight Planes***, with a size of 5x5 cm each. All of this is observed through a ***VRayPhysicalCamera***.



■ **Figure 6.193**
The scene with initial settings.

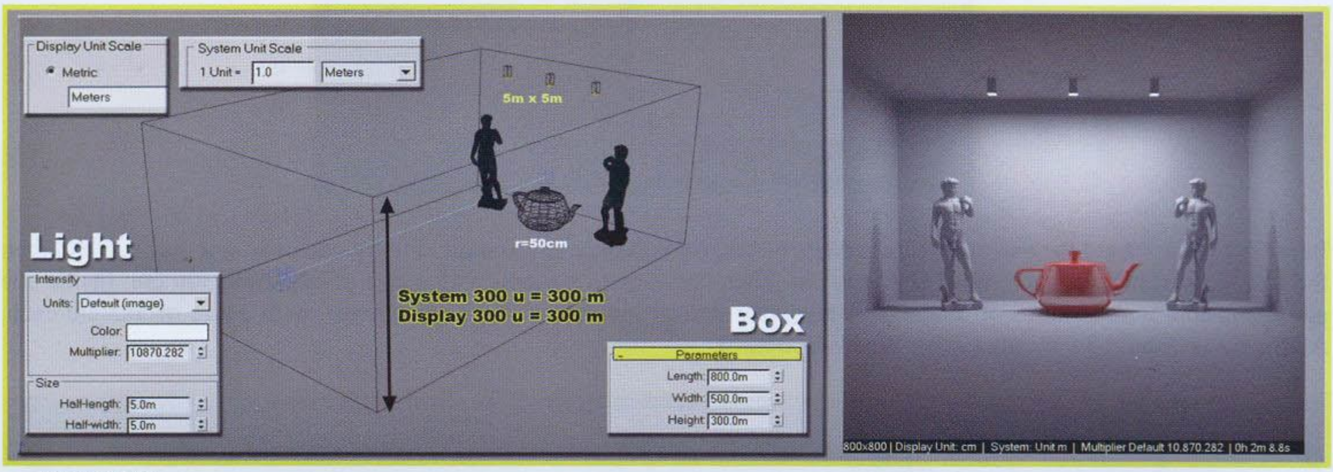
Everything is correctly set. The units of measurement reflect reality and the illumination, seen through a camera objective with physical parameters, is coherent with the scene. At this stage, one changes the Display Unit Scale, from centimeters to meters.



■ **Figure 6.194**
A unit of measurement being used for visualizations different to that of the file. The result is different to the previous one.

In this case nothing has changed at all, apart from the way 3ds Max calls the parameters such as the size of the box or the ***VRayLight***. What has happened is a slight change in visualized units, from cm to m. All the rest is exactly the same, and the result too.

Now try changing the system unit while leaving the ***Display Unit Scale*** in meters. At this point, a consideration must be made. During the production of a render, one should change the value of a system unit always prior to creating geometries. It is not advisable to change the system unit in a pre-existent scene. In fact, in the example, the ***VRayPhysicalCamera*** has undergone a small variation after the variation from cm to m, because of its parameters adapting to the new unit of measurement. In any case, this is the result.



■ **Figure 6.195**
Change in the system unit. The result is almost identical to the previous one.

In this situation, the scene is bigger by two orders of quantity compared to the previous version: the box is now 300 meters high. Illumination is however the same as the previous renders. Let's go back a step. It has been said that by using the default unit, luminosity depends on the size of the ***VRayLight***. It is true that the model is now huge, but so is the ***VRayLight***, which is in proportion to the objects around it. For this reason, even if we decided to scale the whole scene, with the ***Default*** unit nothing would change at all.

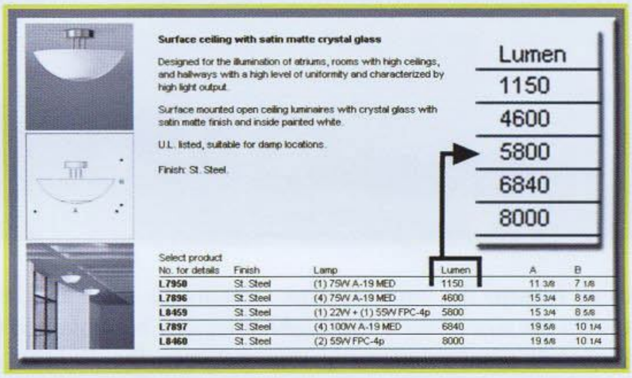
One last point to be made which does not involve *VRay* directly, but can be useful to know. One could decide, for instance, to set both units in meters and model the box by inserting 8x5x3 m instead of 800x500x300 cm. Technically the procedure is correct, but one could have problems in the viewport when one moves too close to a mesh with very few cms of width. By setting the scene in meters, the internal precision of the file changes. With this unit, 3ds Max assumes that one is creating models of a large size, such as an island or a city, not a small room or design object. For this reason, it is advisable to model architecture in centimeters.

LUMINOUS POWER: lumen

The luminous flux expresses the total quantity of luminous energy emitted by a source per unit of time, regardless of light quality and its distribution in space and it is measured in ***lumen***. If we compare light to water, a light source is similar to an open tap and the luminous flux represents the amount of liters which comes out of the tap per unit of time, regardless of its diameter. Te luminous flux is a useful value especially for describing and comparing light bulbs. Those which are usually used in illumination engineering have fluxes which vary from a few hundred ***lumens***, like low-power incandescent light bulbs , to a few hundred thousand lumens of high-power light discharge bulbs, used outdoors.

LIGHT SOURCE	ABSORBED POWER	PRODUCED LUMENS
Incandescent light bulbs	40 W	430 lumen
Incandescent light bulbs	100 W	1,380 lumen
Fluorescent light bulb	18 W	1,300 lumen
Fluorescent light bulb	36 W	3,350 lumen
High-pressure sodium light bulb	400W	47,000 lumen

Take, for example, the catalogue of a famous firm which produces light appliances. This is part of a description of a product.


Surface ceiling with satin matte crystal glass

Designed for the illumination of atriums, rooms with high ceilings, and hallways with a high level of uniformity and characterized by high light output.

Surface mounted open ceiling luminaires with crystal glass with satin matte finish and inside painted white.

U.L. listed, suitable for damp locations.

Finish: St. Steel.

Lumen
1150
4600
5800
6840
8000

Select product

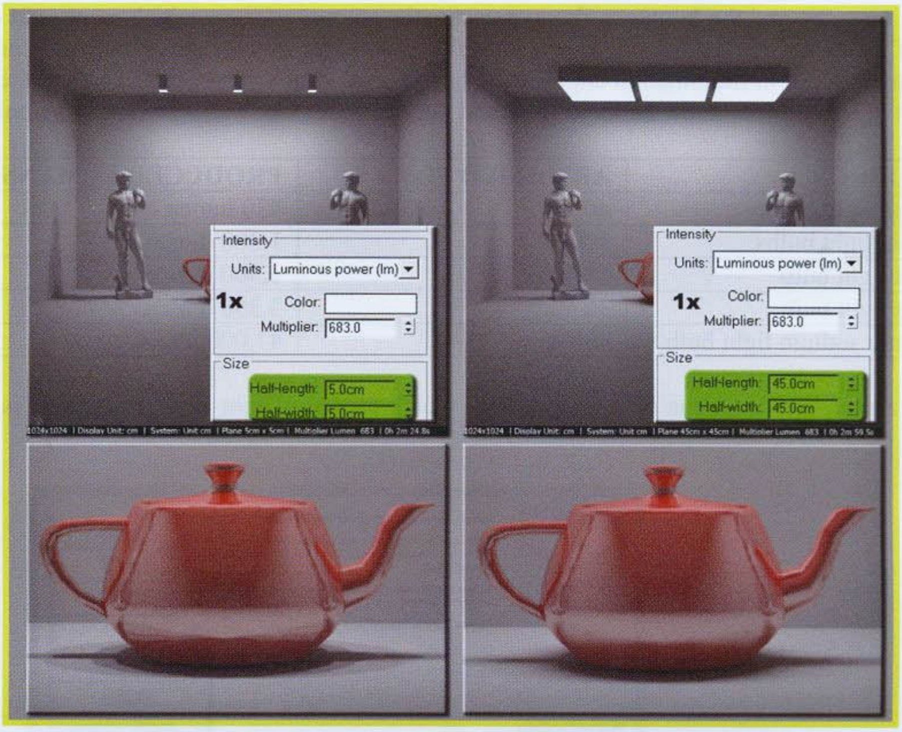
No. for details	Finish	Lamp	Lumen	A	B
L7950	St. Steel	(1) 75W A-19 MED	1150	11 3/8	7 1/8
L7896	St. Steel	(4) 75W A-19 MED	4600	15 3/4	8 5/8
L8459	St. Steel	(1) 22W + (1) 55W FPC-4p	5800	15 3/4	8 5/8
L7897	St. Steel	(4) 100W A-19 MED	6840	19 5/8	10 1/4
L8460	St. Steel	(2) 55W FPC-4p	8000	19 5/8	10 1/4

■ **Figure 6.196**
Extract from a catalogue. Notice how, among size indications and absorbed power values, also data concerning the amount of ***lumens*** produced appear.

In this case it is therefore quite simple to produce a realistic illumination installation with *VRay*. First of all, one must change the ***Multiplier*** parameter and leave the size of the three ***VRayLight Plane*** lights unaltered.

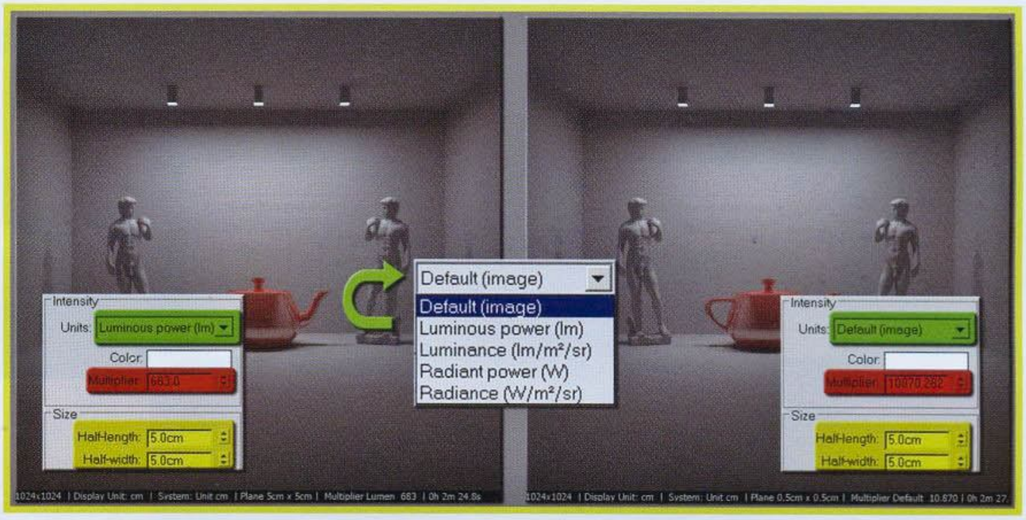
■ **Figure 6.197**
Variation of the ***Multiplier*** of the three ***VRayLights*** with ***lm*** used as a unit of measurements.

If we take a look at the previous image, the increase in ***VRayLight*** intensity causes an increase in the global illumination of the scene. Also, because of the small size of the luminous planes, the shadow produced is quite sharp. One should pay attention to the value 683 lm: it is not a casual value, but as we will see later, it has a precise meaning. In any case, a light source of 683 lm can be compared to an incandescent light bulb of roughly 60W of absorbed power. In the following image we will see what happens if we change the size of the light source.



■ **Figure 6.198**
Change in the size of the three ***VRayLights*** by using the same intensity expressed in ***lumens***. The differences are only in the way shadows are represented.

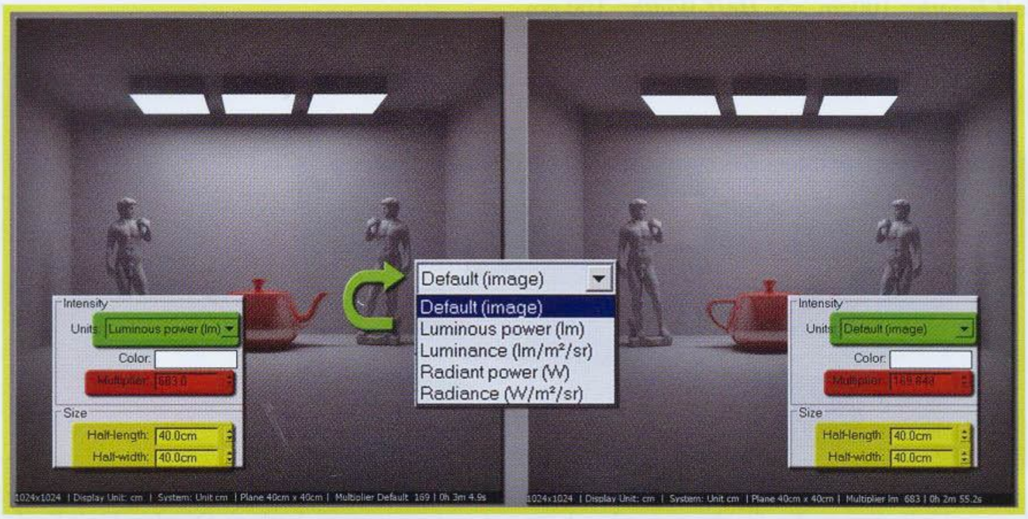
Unlike the ***Default*** unit, by using ***lumens*** the luminous flux remains the same at all times. This way one can change the sharpness of shadows by changing the size of the ***VRayLight***, once one has obtained the right amount of light.
At this point one might ask oneself what relationship runs between the ***Default*** unit and ***Lumens***. ***VRay*** manages internal conversions between different measurements. The user is not required to carry out any calculations. It is, however, useful to understand, at least roughly, how ***VRay*** carries out these operations. We will start off with a scene with settings which have been considered necessary by the user for creating his render: in this case a ***VRayLight*** with a size of 5x5 cm with an intensity of 683 lm. Once the settings have been applied, one changes the ***lm*** parameter to ***Default***.



■ **Figure 6.199**
Change in units of measurement of ***VRayLights*** from ***lm*** to ***Default***.

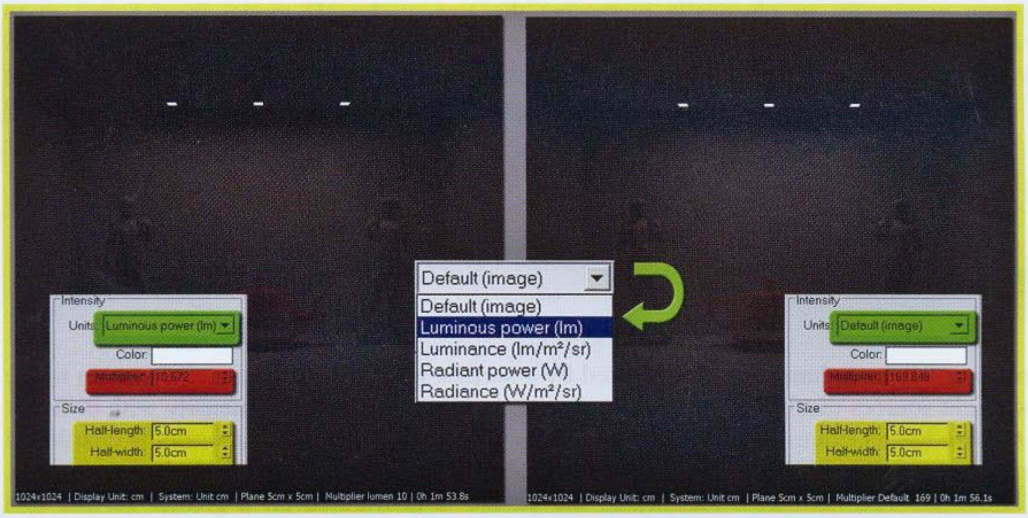
The two renders are identical. ***VRay*** has seen to the conversion from ***lumen*** to ***Default*** units. At this stage one resets the parameters to their initial values, that is 683 lm by automatically changing them with the pull-down menu and one quadruples lumens from 683 to 2,732. Now ***Default*** units will be multiplied by four, reaching 43,480.

Now one must return to the original units, 683 lm and 5x5 cm. This time one changes the size of the ***VRayLights Plane*** to 40x40 cm. The amount of light is always the same and only shadows undergo changes. Convert units from ***lm*** to ***Default***. This time, having increased the length and width of the ***VRayLight*** and as the ***Default*** units depend on their size, ***VRay*** has had to spread the 683 lm on a greater surface, in this particular case, 64 times bigger. So we have gone from 10,870 to 10,870/64 = 169.



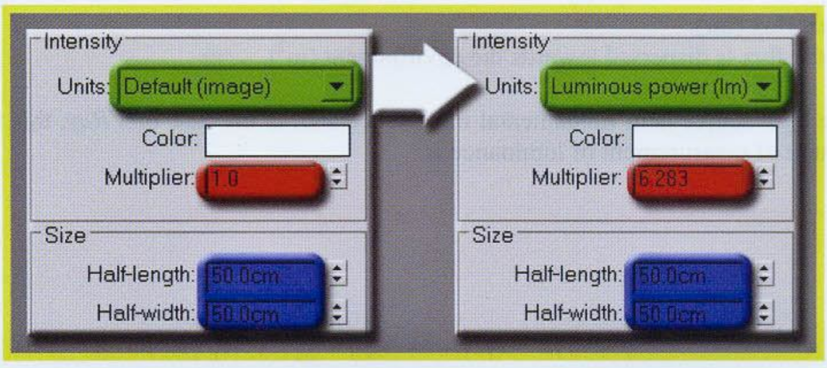
■ **Figure 6.200**
Change in ***VRayLight*** measurements from ***lm*** to ***Default*** units and in the size of the ***Plane***.

Now, with ***Default*** units, ***Multiplier*** = 169 and ***Size*** = 40x40 cm, change the size to 5x5 cm. One should already know what the result of the render will be: darker, because of the smaller emission surface. If after one converts the ***Default*** units to ***lumen*** with the ***VRayLight*** = 5x5 cm, the result is no longer 683 lm, but only 10 lm.



■ **Figure 6.201**
Passage from ***Default*** to ***lumen***.

Which calculations does ***VRay*** carry out in order to transform ***Default*** units to ***lm***? The answer is quite simple and one can observe them through a simple exercise. Create a ***VRayLight Plane*** of 50x50 cm, that is, a light with a surface of 1m^2. Set the unit of measurement to ***Default*** = 1. Finally, convert from ***Default*** to ***lumens***. The result is a familiar number: 6.283. It is the number 2π.



■ **Figure 6.202**
Conversion from ***Default*** units to ***lm***.

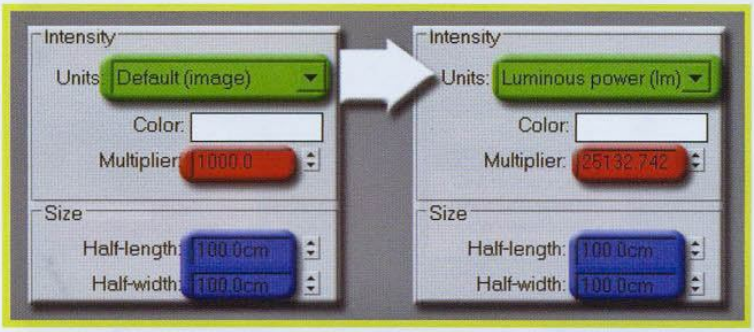
As a result, the formula for converting from ***Default*** units to ***lm*** units is the following:

Default unit x VRayLight Area (m) x 2п = lumen units

For example:

VRyaLight Plane => ***Half-length***= 100cm => ***Half-Width***= 100 cm
Multiplier Default => 1,000

1,000 x 2m x 2m x 2π = 25,132 lm



■ **Figure 6.203**
Practical example of conversion from ***Default*** units to ***lm*** units.

One last consideration on System Units Setup. So far the scene has been set to centimeters, both as Display Unit Scale and also as System Units Setup. The change in Display Units Scale, as always, causes changes exclusively in the way 3ds Max shows the user the units of the parameters, without affecting renders or anything else. Unlike ***Default*** units (which have the main characteristic of having the ***Multiplier*** directly and proportionally connected to the size of the ***VRayLight***), ***lumens*** do not depend on the area of the ***VRayLight***. In fact, by changing the ***System Units Setup*** from cm to m, as we have already seen previously, or physically increasing the size of the 3D model from 5x8x3 m to 500x800x300 cm, the situation would change completely. In this last case, the ***VRayLight***, which would then be of 5x5 m and no longer 5x5 cm, would still emit 1,000 lm in any case, which would prove to be too little for a room of that size. One would therefore have to not use 1,000 lm anymore, but instead 1,000,000 lm, in other words, 1,000 lmx100x100.

Luminance: lm / m² / sr

Luminance expresses the amount of light or luminous flux emitted by a surface in a given direction, in relation to the size of the surface itself. Two light sources can have, in the same direction, the same luminous intensity with different luminance values. If one of the two has an extension which is a lot greater than the other one, the intensity is distributed on a greater surface and therefore has a lower density. If we place a frosted glass globe around a light bulb or candle, the whole globe becomes luminous. The luminous intensity of the candle is distributed on a greater surface and therefore luminance is decreased. The glass sphere therefore has a smaller amount of luminance, at equal luminous flux. One can then understand how this quantity is used to determine the vision and sensation of dazzling brightness. Luminance can be equally referred to surfaces that emit light, as well as geometries which reflect it.

From the definition, one highlights the fact that there are two main factors of luminance:

.The total amount of luminous flux (***lm***).
.The surface through which the flux is dispersed towards the environment (m^2).

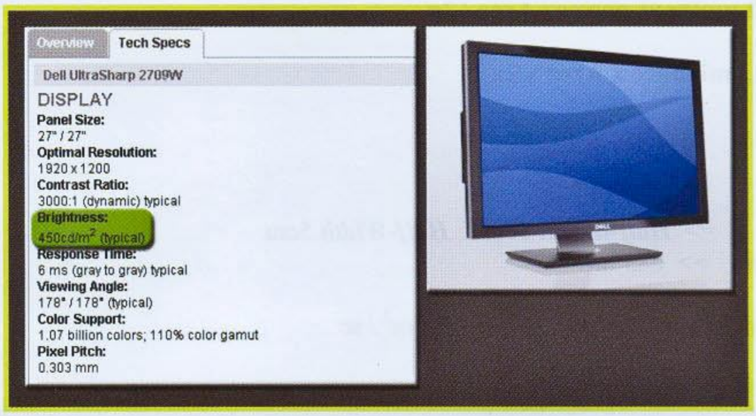
It is clear that the size of the light source is of fundamental importance and, in the case of ***VRay***, this translates into the size of the ***VRayLight***. The unit of measurement of luminance is:

Luminance = lm / m² / sr

Or

Luminance = cd/m²

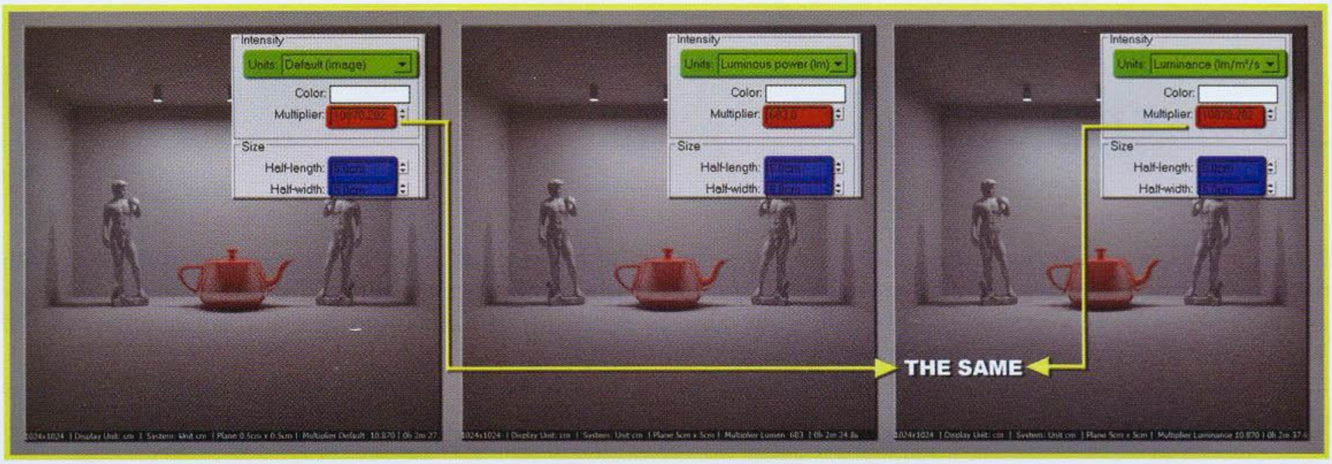
A classic example of the use of this unit is the luminance indicated in LCD or Plasma televisions, of which it is possible, by observing technical specifications, to know exactly the size of the luminous panel and luminance.



■ **Figure 6.204**
Indication of ***luminance*** and ***size*** of the luminous source of a LCD panel.

Observing this technical plan it is possible to obtain precious information about the light source. The first part concerns size, 80x50x100 cm. The second part regards luminosity, or rather, luminance of the LCD panel. Regarding ***lumens***, the difference lies in the fact that the unit of measurement is closely connected to the size of the light source.
In order to understand this concept better, imagine a small spotlight, with a size of 5x5 cm. The unit of measurement is the lumen, so, regardless of the size of the ***VRayLight Plane***, it will emit the same amount of light all the time, say 1,000 lm. The variation in the size of the ***Plane*** will only influence the smoothness of the shadow. Now set the unit to ***Luminance***. The light, which has a size of 5x5 cm, according to the technical plans, emits 1600 cd/m^2. This unit is closely connected to the size of the light source and one can see how, for a correct representation, it is not sufficient to insert the ***Multiplier***, but it is necessary to insert also the correct ***Plane*** size. If this is not so, the light source emits more or less light according to technical data.

An observation must be made concerning the units and conversion between ***Default, Luminous power*** and ***Luminance***. Set the scene to ***lm*** units and a ***Plane*** source of 5x5 cm. Convert the ***lm*** both to ***Default*** and ***Luminance*** units.

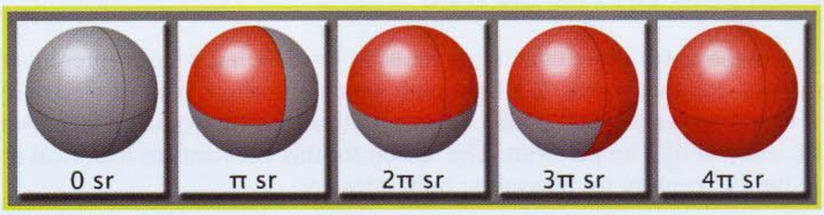


■ **Figure 6.205**
Conversion between ***Luminous power***, ***Default*** and ***Luminance***.

The ***Multiplier, Default*** and ***Luminance*** are the same. This means that also the behavior of ***VRayLights*** is, both as a function of the surface and the sharpness of shadows. So:

Luminance = lumen per square meter per semisphere => lm / m^2 / sr

Where 1 semisphere is equivalent to 2π steradians.



■ **Figure 6.206**
Graphic representation of steradians.

The conversion from ***Luminous power*** (***lm***) to ***Luminance*** is now clear:

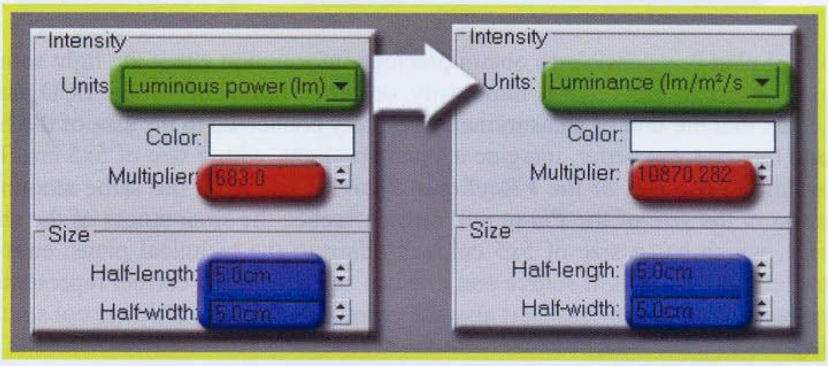
Luminance (lm/m2/sr) = Luminous power / Area / 2π

Luminous power (lm) = Luminance x Area x 2π

For example:

VRyaLight Plane => ***Half-lenght*** 5cm x ***Half-Width*** 5cm
Multiplier Luminous power => 683 lm

Luminance = 683 lm / 0.1m / 0.1m / 2π = 10,870 lm / m^2 / sr



■ **Figure 6.207**
Practical example of conversion between ***Luminous power*** and ***Luminance***.

As for the behavior of ***VRayLights***, refer to the explanation regarding the ***Default*** units, as they are identical and depend on the size of the ***VRayLight***.

RADIANT POWER: Watt

Radiant power or radiant flux is the radiometric correspondent of luminous flux which is represented in ***lumens***. With radiant power one indicates the total power irradiated by an object in all directions per unit of time. The measurement unit is the ***Watt*** (Joule x second). A light bulb which absorbs 100 Watts of electric current has a radiant power of 100 Watt, but not all these Watts are transformed to energy.
In order to understand the concept of radiant power better, it is necessary to introduce the concept of luminous efficiency.

We have seen that there is a photometric reference in all radiometric units. Their accordance is defined by the luminous efficiency η (lumen/Watt), as a fraction of radiant power which lies within the visible band.

Efficiency = luminous flux / radiant power

η = lm /Watt

In other words, the luminous flux is equal to the radiant energy (Watt) x 683 (lm/Watt)

Luminous power (lm) = Radiant power (W) x 683 (lm/Watt)

The 683 factor (lm/Watt) is due to the sensitivity of the of the retina at λ = 540 nm, which represents the maximum peak of the curve of scotopic sensitivity. In other words, if a light bulb which absorbs 1 Watt of electricity emanates 1 Watt at a wavelength of 540 nm, it would produce a total of 683 lumens. In this case we would be faced with a source with an efficiency of 1, that is, an efficiency of 683 lm per Watt. The reason for this constant are historical and date back to the measurement samples used in the 19th century for measuring light intensity.

Incandescent light bulb: these are the common light bulbs invented by Edison. In these lamps only 5-15% of absorbed power becomes light. The rest is dispersed in heat. They have a low efficiency equal to about 13 lm/W. They also have a limited lifetime: 1,000 hours. The only advantage of these bulbs is the low cost, but only apparently: these bulbs consume a lot of energy, much more than the light they produce.

Halogen lamp: these have incandescent bulbs but they have a luminous efficiency between 13 and 22 lumen/Watt. They have a lifetime of roughly 2,000 hours, about twice as much as a normal light bulb. However they also emit a greater amount of ultraviolet rays. It is therefore advisable to place these lamps at least one meter away. Alternatively one can shield it with a glass pane. Halogen lamps are used when a beam of light which is well localized is necessary, for example to illuminate a desk or an angle of the house.

Fluorescent tubes: these are tubes which contain gas, such as mercury vapor, argon or neon and are dressed in fluorescent substances. The light emitted by these bulbs is different to natural light and can tire ones eyes. Also the nervous system can suffer because of the flicker which is perceived by the optical nerve. There are however advantages: high efficiency of 90 lumens/Watt, average lifetime of 10,000 hours, cheap price and variety of luminous power offered. They are appropriate for large spaces.

Compact fluorescent light bulbs or energy saving bulbs: they have a low level of consumption and a long lifetime which decreases if they are turned on more than once an hour. Their luminous efficiency varies from 50 to 75 lumen/Watt, allowing a considerable reduction of energy consumption compared to incandescent light bulbs with the same luminous flux. It is enough to think that a 20 Watt compact fluorescent bulb provides the same performance as a normal 100 Watt lamp. Compared to the latter, however, a fluorescent compact bulb has a price which is sometimes 10 to 15 times as much, but its lifetime is 10,000 hours. The initial cost is thus amortized, because the energy saving lamp consumes less and lasts longer.

We shall now give a practical example: a 150 Watt incandescent bulb emits about 2,000 lumens. Its efficiency is equivalent to 13 lumen/Watt. If, for instance, the emission had been 102,450 lm, this would have meant that this bulb were emitting all its radiant power in the visible band. In reality this is not possible, because an incandescent lamp produces mostly heat and not much light. Going back to the subject of luminous efficiency, in order to know how many Watts are transformed into light, it is enough to divide the total number of lumen produced by the source by the constant 683 lm/Watt:

Luminous power (lm) / 683 (lm/Watt) = Radiant power within the visible band (Watt)

2,000 (lm) / 683 (lm/Watt) = 2.9 Watt.

FLOURESCENT LIGHT BULB: SPECIFICATIONS

Model	Temp	Watts abd	Lumen	Luminous Efficiency
ESL7	Cool White	7 Watts	320 Lm	46 Lm/W
ESL11	Cool White	11 Watts	550 Lm	50 Lm/W
ESL7W	Warm White	7 Watts	320 Lm	46 Lm/W
ESL11W	Warm White	11 Watts	550 Lm	50 Lm/W

INCANDESCENT LIGHT BULB: SPECIFICATIONS

Watt absorbed	Lumen estimated
40	> 450
60	> 800
75	> 1,100
100	> 1,600
150	> 2,600

■ **Figure 6.208**
Example of ***Watt***, ***Lumen*** and ***luminous efficiency*** parameters being used.

In order to obtain luminous efficiency, that is, the percentage of Watts which from absorbed energy become luminous flux, one must divide the "visible" Watts by the absorbed ones. In the previous case:

Luminous efficiency = ("Visible" Watts / Absorbed Watts) x 100

1.9% = (2.9 / 150) x 100

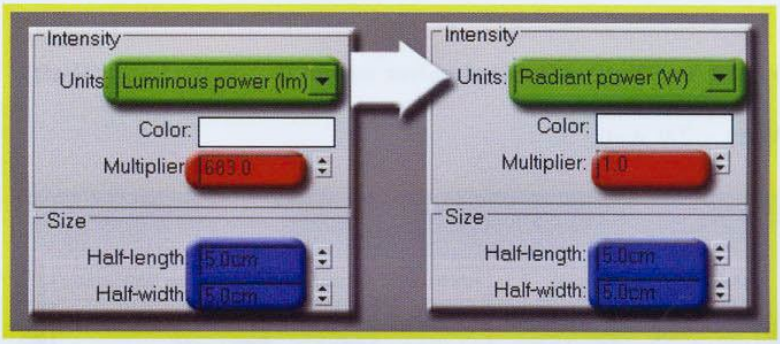
Category	Type	Luminous efficiency (lm/W)	Luminous effectiveness (%)
Combustion	Candle	0.3	0.04 %
Incandescence	5 W Tungsten	5	0.7 %
	40 W Tungsten	12.6	1.9 %
	100 W Tungsten	17.5	2.6 %
Fluorescent	5-24 W compact fluorescent	45 - 60	6.6 - 8.8 %
	34 W fluorescent (T12)	50	7 %
	32 W fluorescent (T8)	50	9 %
	36 W fluorescent (T8)	>93	> 14 %
	28 W fluorescent (T5)	104	15.2 %
Gas	400 W high-pressure sodium	140	20.5 %
	Ideal source	683	100 %

At this stage, after a brief introduction on radiant power, it is now time to how *VRay* interprets this unit of measure. First of all, as radiant power is the corresponding radiometric measure for luminous flux, measured in ***lumen***, one can expect that also in this case, the size of the ***VRayLight*** does not affect the amount of light produced, but only on the type of shadow. Just like ***lumen***, whether the ***VRayLight Plane*** is 5x5 cm or 50x50 cm, the luminous flux remains the same and only affects the sharpness of shadows.
As far as conversion is concerned:

683 lm = 1 Watt of radiant power (in the visible band)

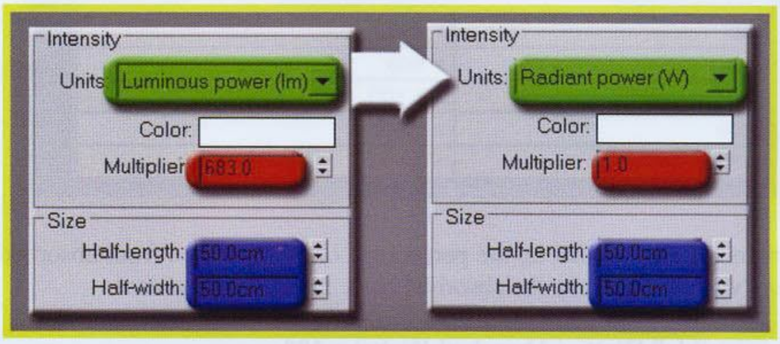
The Watts defined through the unit of measurement ***Radiant power*** of ***VRayLights*** does not represent the absorbed power, but the part which is transformed into radiant power with a wavelength between 300 nm and 750 nm, the part which can be seen by the human eye.

By observing the behavior of *VRay* one understand precisely what is happening at the time of conversion.



■ **Figure 6.209**
Practical example of conversion between ***Luminous power*** and ***Radiant power***.

The conversion between ***lumen*** and ***Watt*** values corresponds to the conversion coefficient of 683 lm/Watt which we have already encountered previously. If one changes the size of the ***VRayLight Plane***, by using these two units of measurement, no changes are applied to the ***Multiplier***.



■ **Figure 6.210**
Practical example of conversion between ***Luminous power*** and ***Radiant power***. The modification of the size of the ***VRayLight*** does not create any variation in luminous intensity.

RADIANCE: Watt / m^2 / sr

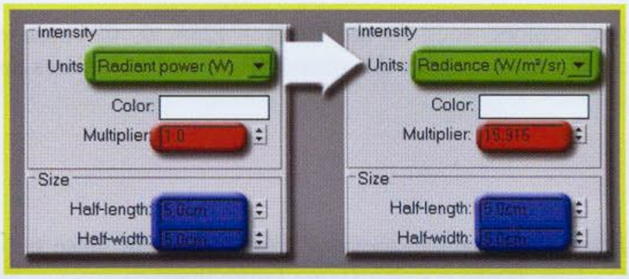
Radiance is the radiometric unit which corresponds to luminance. All the observations made on the behavior of or radiant power corresponding to luminous flux are valid here as well. The most important point is that the light produced by the ***VRayLight*** is closely connected, at equal ***Multiplier***, to its size.

To convert radiant power to radiance, the formula is the same one used to convert luminous power into luminance.

Radiance = Radiant power / Area / 2π

Radiant power = Radiance x Area x 2π

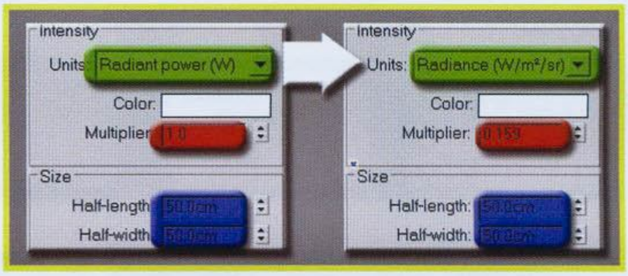
Observe the scene used so far.



■ **Figure 6.211**
Practical example of conversion between ***Radiant power*** and ***Radiance***.

Radiance = 1 W / 0.01m^2 / 6.28 = 15.9 W / m^2 / sr

Obviously by increasing the size of the ***VRayLight*** when the unit is expressed in ***Radiant power***, by converting it after in ***Radiance*** the ***Multiplier*** decreases, in order to adapt to the new ***Plane*** size.



■ **Figure 6.212**
Practical example of conversion between ***Radiant power*** and ***radiance*** with ***VRayLight*** size having been changed.

NOTE

One can consider this complex chapter concerning units of measurement within ***VRayLights*** over. In the previous examples, ***Planes*** have been used for the simple fact of it being easier to calculate. In the case of spherical ***VRayLights*** being used, it is enough to use the formulas of the area of a sphere instead of using that of a plane.

Area sphere = $4\pi R^2$

PS: In the ***1.5 FINAL*** version of ***VRay*** some important calculation variations have been introduced, for conversion between various units, also adding two new options to the ***VRay: Global switches*** rollout. These are the ***Legacy Sun/Sky//camera models*** and ***Use 3ds Max photometric scale*** parameters. Thanks to this last parameter, ***VRay*** is able to produce a photometric illumination which is the same as that of 3ds Max. In fact, up until this moment ***VRay*** did not carry out completely physically accurate calculations. In order to maintain compatibility with old units of measurement of files prior to version 1.5 one must use the ***Legacy Sun/Sky/camera models*** parameter (although with a few variations in unit conversion), otherwise new files have the ***Use 3ds Max photometric scale*** option active by default. For this reason, the examples of the previous chapter may not correspond to the files carried out with ***v1.5 FINAL***.

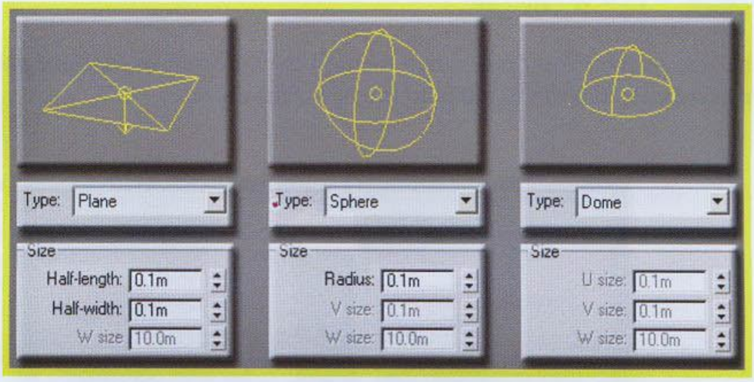
Color - it is possible to specify the color of light emitted by the ***VRayLight***.

■ **Figure 6.213**
Use of three RGB colors in different situations and combinations.

Notice how the use of "pure" colors, such as red RGB = [255, 0, 0] cause obvious burns, whereas using it in combination with other color sources attenuates this phenomenon and neutralizes dominant colors.

Size

Size - as we have already anticipated in the previous examples, with these options we can specify the size of the ***VRayLight***, both spherical or plane. If one is using ***Dome Light***, these parameters become invisible.

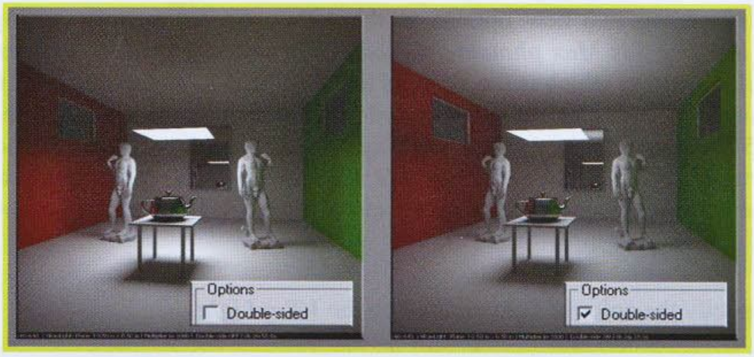


■ **Figure 6.214**
The ***size*** parameter and the three types of ***VRayLights***.

By using the three types of sources, ***Size*** parameters automatically change to indicate the correct size, length and width for ***VRayLights Plane*** and radius for ***VRayLights Sphere***. ***Dome Light*** has no options. In order to understand the effects of the ***Size*** parameters, one can cover this subject in the previous pages.

Options

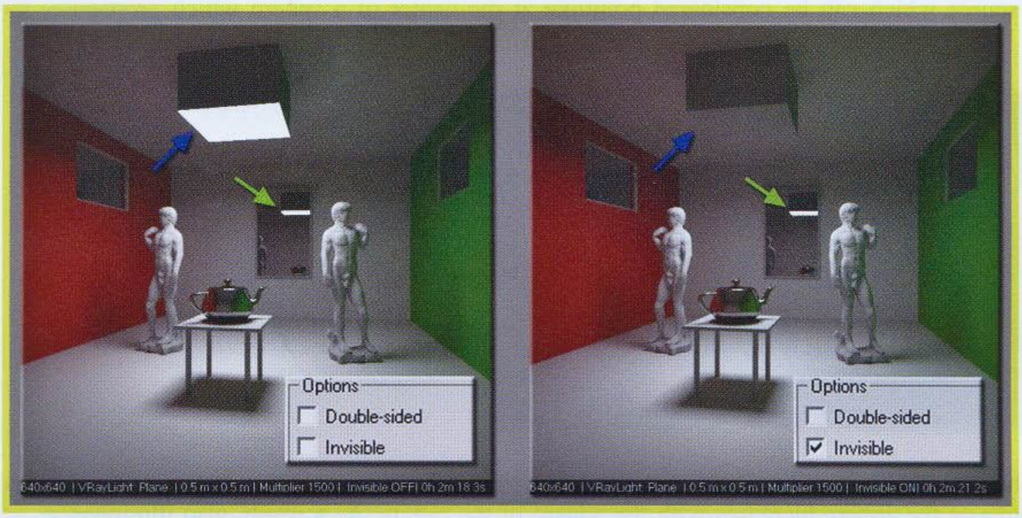
Double sided - default ***VRayLights Plane*** emit light only in the direction indicated by the arrow in the icon. By activating the ***Double-sided*** option one can emit a luminous flux from both sides. The activation of the ***Double-sided*** parameter has an effect only on ***VRayLights Plane***.



■ **Figure 6.215**
Toggling the activation of the ***Double-sided*** parameter.

After the activation of the ***Double-sided*** parameter, the ceiling becomes illuminated by the upper part of the ***VRayPlane*** which is turned off by default.

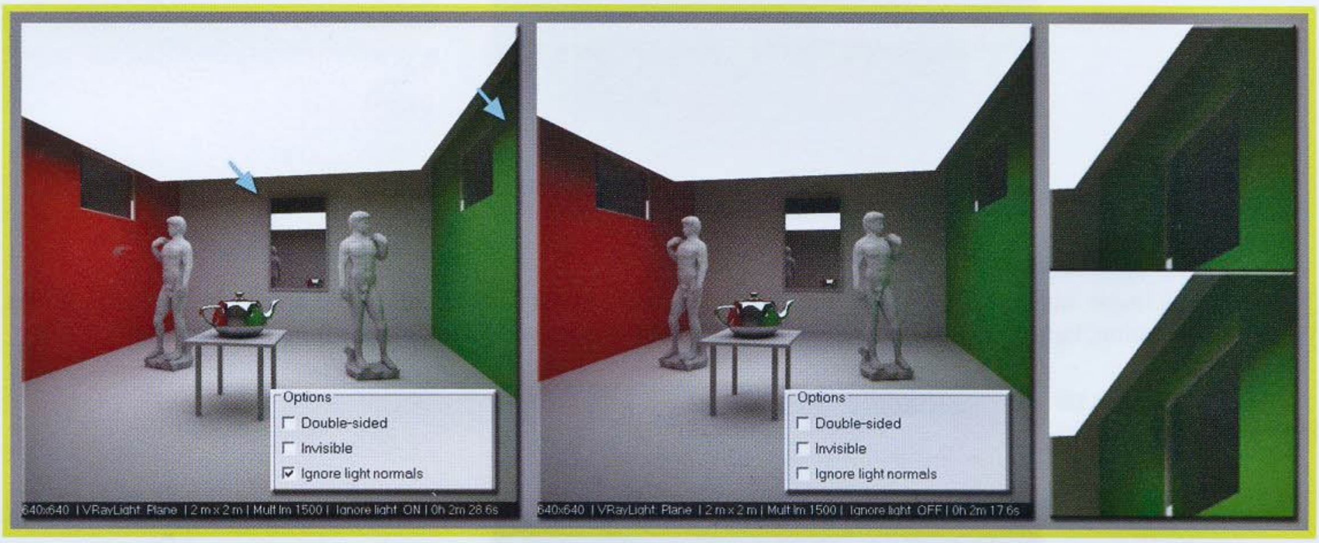
Invisible - this parameter controls visibility of ***VRayLights***. By activating it, ***VRayLight*** continue to emit light while maintaining their properties of intensity and color, but the plane or sphere are not rendered. Notice how by visibility one means the direct vision of lights through the camera. If the luminous sources were visible on reflective objects, the ***Invisible*** option would have no effect. In order to make them invisible also on chromed surfaces, one must use the ***Affect specular*** option, a parameter which we will cover shortly.



■ **Figure 6.216**
By activating the invisibility of the ***VRayLight*** means that it will not be rendered if it is observed directly through the camera.

By activating the invisibility of the ***VRayLight***, the ***Plane*** disappears from direct view, but is visible in reflective objects, as one can see from the mirror and the metal teapot.

Ignore light normals - ***VRayLights*** produce a luminous flux of equal intensity in all directions of emission. If the ***Ignore light normals*** is deactivated, the luminous flux is more intense in directions which are normals of the surface.

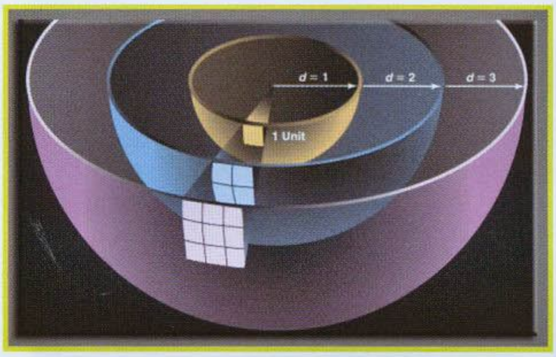


■ **Figure 6.217**
Activation of the ***Ignore light normals*** option. Its effect is minimal, but in any case noticeable.

The difference between renders with the ***Ignore Light normals*** active (by ***default***) and not, is easily visible. By observing the previous test carefully, one can see a first difference. If we look at the shadow produced on the vertical walls, it is much sharper with the ***Ignore light normals*** option active, instead by deactivating the option it is smoother. Another observation on general luminosity is that in default circumstances, luminosity is slightly greater than when the parameter is turned off. This can be explained because of the simple fact that with ***Ignore light normals*** active, the plane emits the same amount of light from all the points on its surface, whether they are nearer the center or more external. This means that even the furthest spot of the ***VRayLight Plane*** has an intensity which is equivalent to the central point. Therefore a sharper shadow is generated.

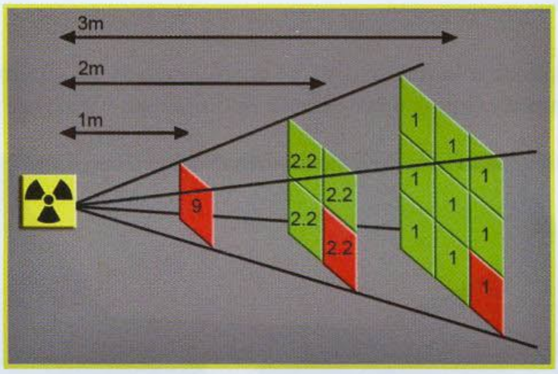
If this dos not happen, a situation which can be obtained by deactivating ***Ignore Light normals***, the plane emits more light from the center and less from the edges. This provides a smoother shadow.

No decay - it is common experience to notice how the intensity of light decreases as we draw further away from the source. For example observe how the light from a headlight of a car attenuates as it becomes further away from the headlight itself. Light behaves in a predictable and measurable way and can be described by a mathematical formula, called inverse square law.
This law is applied to fields that propagate in a homogeneous way in all directions like light, the magnetic field and the gravitational field. The bodied which are able to produce fields of this type are called pointlike sources. The following image shows a radioactive source which can be considered pointlike at the center of a sphere which propagates its radiation in all directions.



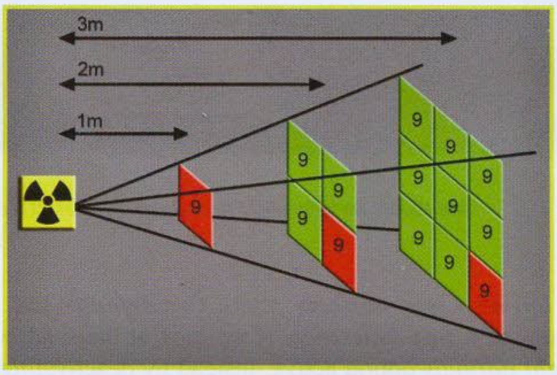
■ **Figure 6.218**
As the distance from the geometric center increases, the surface of the sphere increases.

From this, one can deduce that the total power irradiated by a source in every direction, called radiant power, remains constant, whereas the total area of the sphere increases with the square of the radius. As a result, also illuminance, or generally radiation emission, varies with the inverse square of the distance from the source.



■ **Figure 6.219**
Quadratic variation of intensity of radiation emission.

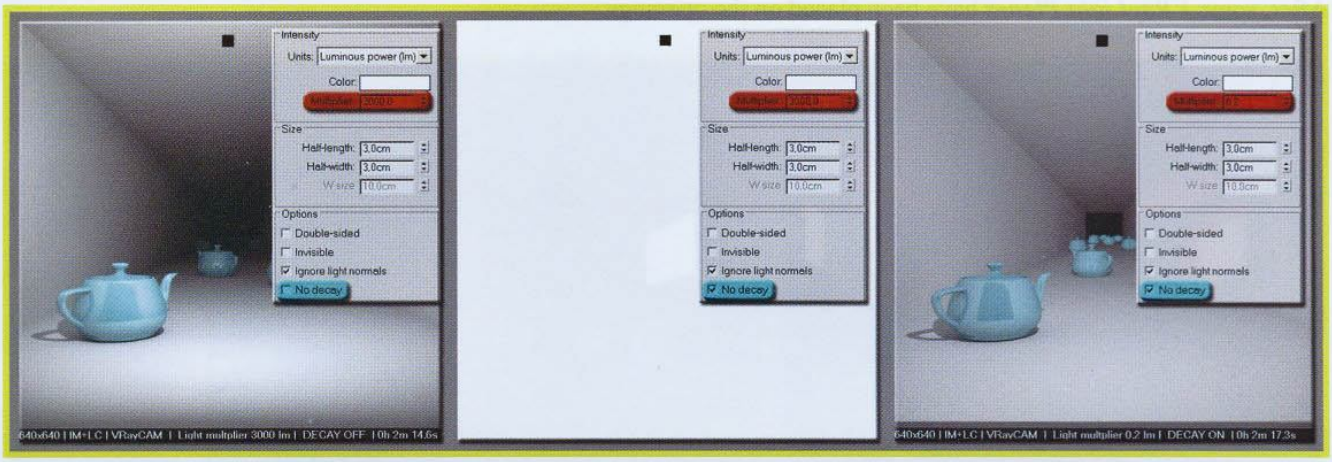
If behavior was linear, impossible in nature, the amount of radiation per surface unit would always be the same. In this hypothetical situation, light would have the same intensity both at 1 cm and 1 km distance from the source.



■ **Figure 6.220**
No variation of intensity of radiation emission.

So one can say that for a physically correct representation of the luminous flux, one is obliged to use a system with a quadratic light decay.

By default *VRay* uses quadratic decay for light propagation. By activating the ***No decay*** option, one forces the ***Chaosgroup*** product to not follow this law.



■ **Figure 6.221**
Example of the use of decay.

In the left-hand image, a small light with a size of 3x3 cm, placed above the observation point, emits a luminous flux of 3,000 lm. As it is correct to expect, when there is decay, light tends to decrease as one draws further from the source. By deactivating the decay, that is, by activating the ***No decay*** option, the 3,000 lm which are emitted do not die out with square distance but propagate forever. 3,000 lm is a very high value and without using decay, it is necessary to decrease the ***Multiplier*** in order to avoid overexposure. With ***Decay*** deactivated there is no decrease in luminous flux and light is of equal intensity in any point along the corridor.

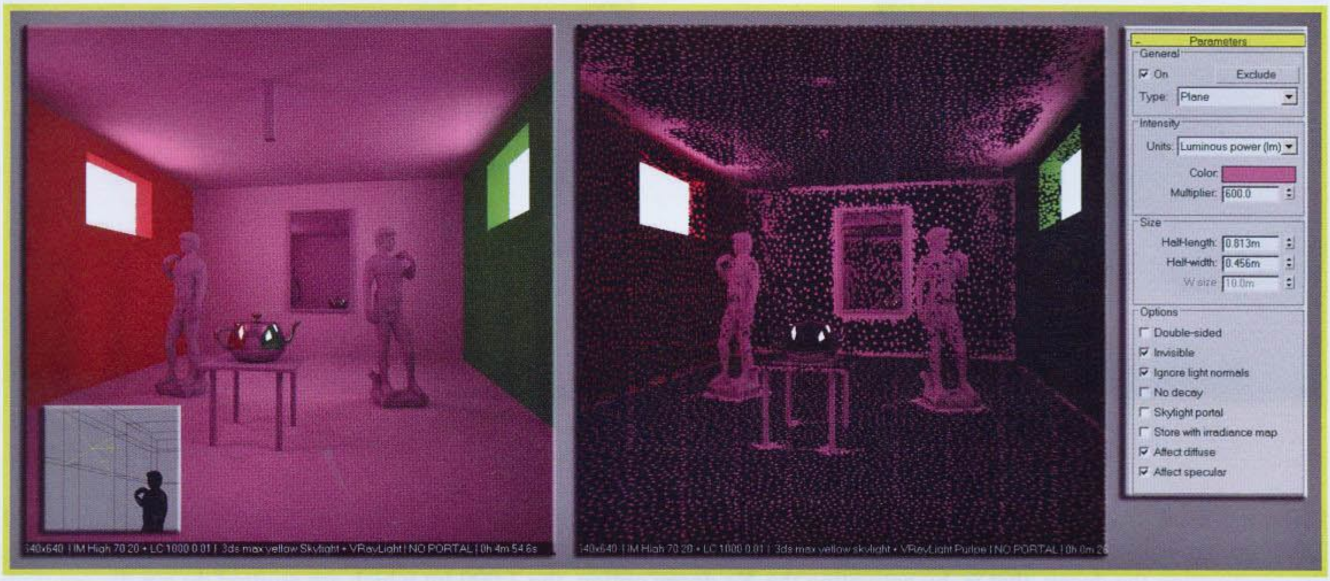
Skylight portal - when active, the ***Color*** and ***Multiplier*** parameters of ***VRayLights*** are managed by the ***Environment*** of *VRay*, also called ***skylight***, located in the Renderer panel in the ***VRay: Environment*** rollout.
This option is usually used only for internal renders. The classic procedure consists of positioning some ***VRayLights*** of the ***Plane*** type on windows of the same size as the openings. Once placed, they can be used as "generators of Environment". This makes it easier to allow the light from the sky cap to flow into the spaces. By creating a few ***VRayLights*** which "replace" the ***skylight*** of the ***Environment*** one can obtain a better result in less time.

In order to understand the behavior of this parameter, observe the following parameters. It is the same model used previously, illuminated only with the Environment of 3ds Max.

■ **Figure 6.222**
Rendering process of an indoor environment with the 3ds Max Environment.

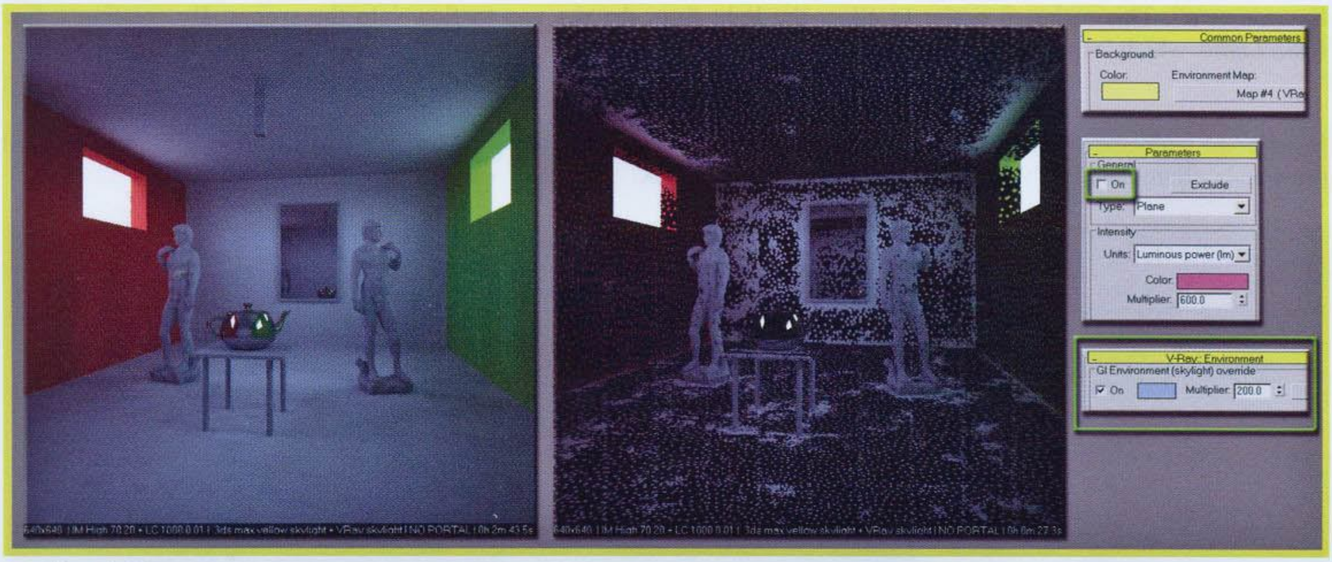
In this scene the only light source is the 3ds Max Environment. A ***VRayColor*** map with a very high RGB value has been used, because one is using a ***VRayPhysicalCamera*** as the observation tool, with exposure control active.

The parameters used for the GI are medium-high. This ensures good quality shadows, as well as a good quality Global Illumination. By using ***skylight*** alone, shadows are calculated by using ***IM*** samples only. As one can see from samples, shadows are recognized, for example under the table, but they are not very precise.
After, some ***VRayLight Planes*** have been placed on the openings.



■ **Figure 6.223**
VRayLight Planes being used as light sources on the openings. The ***Skylight*** portal is deactivated, as by default.

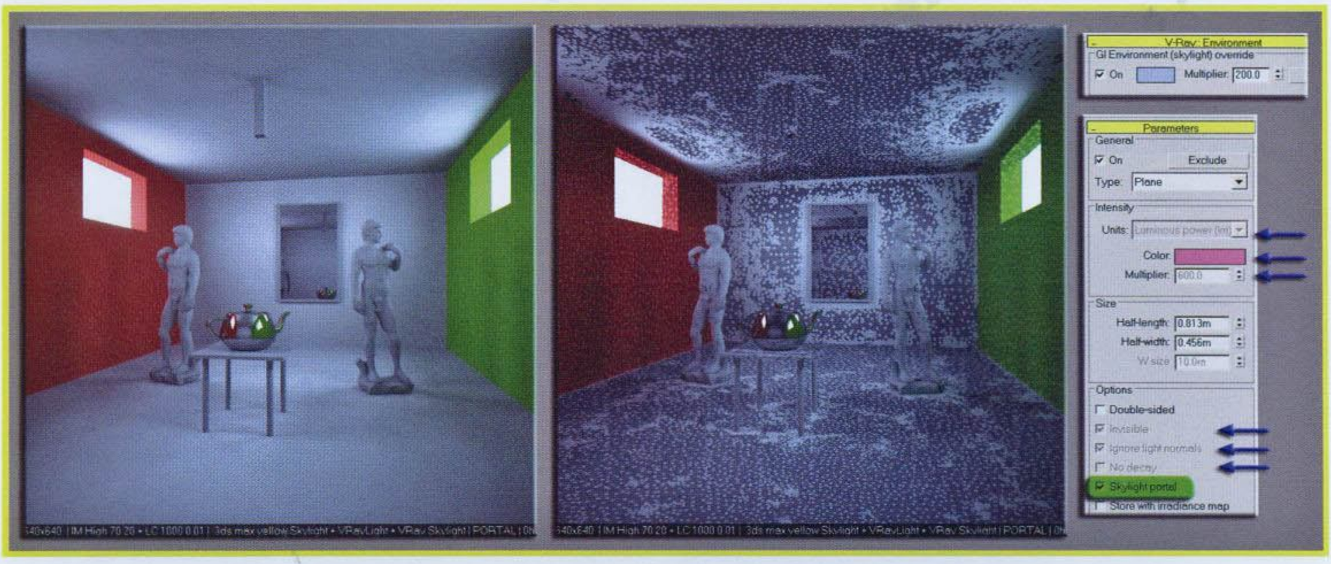
By using ***VRayLights***, the effect of the 3ds Max Environment is missing. This is because the lights of ***VRay***, also with ***Invisible*** option active, obstruct the passage of the 3ds Max Environment. Notice how there is an improvement both in the GI samples and also in the quality of shadows which, in this case, are not calculated via ***IM*** samples, but in a direct manner via the ***QMC*** system of ***VRayLights***. This subject will be deepened later on. At this point the ***VRay skylight*** is activated and ***VRayLight Planes*** are deactivated.



■ **Figure 6.224**
VRay ***skylights*** being used.

In this case, illumination is very similar to the use of the 3ds Max Environment, with shadows not very defined, and messier samples compared to the use of ***VRayLights*** only. Notice the absence of the GI, due to the Background of 3ds Max and the deactivation of the ***Plane*** lights of ***VRay***.

All that remains to be done is to deactivate the ***VRayLights***, with the parameter ***Skylight portal***.



■ **Figure 6.225**
Activation of the ***Skylight portal***.

The first thing that can be seen after activating the ***Skylight*** portal is that the following values of *VRay* are grayed out:

. ***Units***
. ***Color***
. ***Multiplier***
. ***Invisibile***
. ***Ignore light normals***
. ***No decay***

The reason for this is that ***VRayLight*** is none other than a *VRay* extension, having the asset of generating shadows through a direct system. Both the color and the intensity are managed directly within the ***VRay: Environment*** rollout. Also, thanks to the ***VRayLights***, shadows are much more defined and have a higher quality.

Store with irradiance map - when this parameter is activated and GI is calculated through the ***Irradiance Map***, *VRay* computes shadows during the generation of the Global Illumination itself and stores the information within the ***Irradiance Map***. This makes the GI calculation slower, but the final render is much faster, especially if there are many lights. As one can save the ***IM***, it is possible to render animations very fast, by taking the data for shadows from the ***Irradiance Map***.

The following image represents an indoor scene illuminated by nine ***VRayLights Plane***, with the ***Store with irradiance map*** deactivated. The GI has been calculated with a ***IM+LC*** method with medium-high values. The ***IM*** has been set with the ***High preset*** and with ***Subdivs*** = 70, and ***Interp. samples*** = 15, whereas the ***LC*** has been set with ***Subdivs*** = 800 and ***Size*** = 0.01. The calculation of the scene has been carried out in two passages. With the first, only the GI has been calculated in order to save both the ***Light Cache*** and the ***Irradiance Map***. This has been made possible by activating the ***Don't render final image*** parameter located in the ***VRay: Global switches*** rollout. With the second passage the maps which have just been generated have been saved in the memory, by deactivating the ***Don't render final image*** option and activating ***antialiasing***, thus obtaining the final image. The observations we will make concern the GI and shadow quality and rendering time during the two passages, comparing them with the ***Store with irradiance map*** parameter on and off.

■ **Figure 6.226**
Calculation of the ***Irradiance Map*** with the ***VRayLights*** option, ***Store with irradiance map*** deactivated.

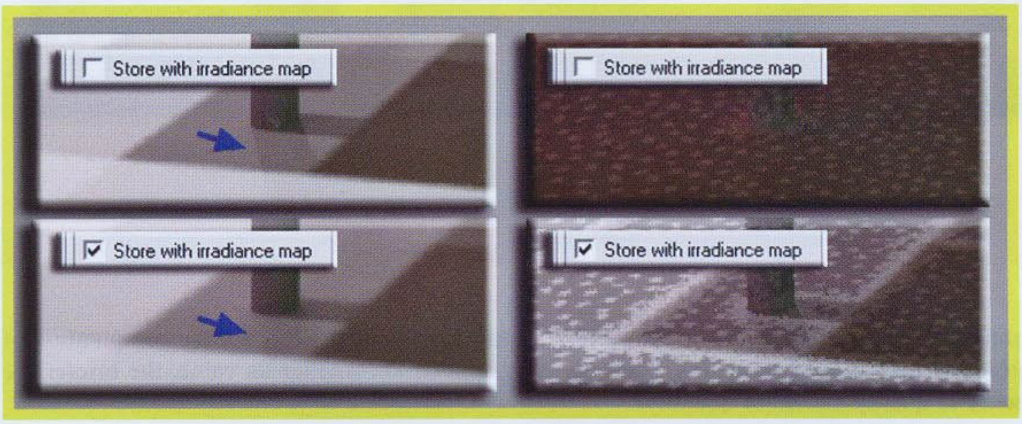
The time taken for GI calculation is about 4 min. 30 sec. Notice the distribution of samples, placed in a much more orderly manner and with a greater presence in areas where geometrically there is a greater need, like edges or organic surfaces. As far as calculation time of the final render with shadows, colors and reflections applied is concerned, 2 min. 30 sec. have been required. Up to this point, nothing unusual.

At this point we shall repeat the same tests, with the same identical parameters, but with the option ***Store with irradiance map*** active.

■ **Figure 6.227**
Calculation of the ***Irradiance Map*** with the ***VRayLights*** option, ***Store with irradiance map*** active.

The differences between the two images are basically two. The first concerns rendering time and the second concerns quantity and distribution of samples. By activating the ***Store with irradiance map*** option, while calculating the GI via the ***IM***, ***VRay*** takes into account not only geometries, but also the shadows projected by the ***VRayLights***. This makes ***IM*** calculation slower but precious minutes are gained during the final render, as they have already been inserted into the ***IM***.

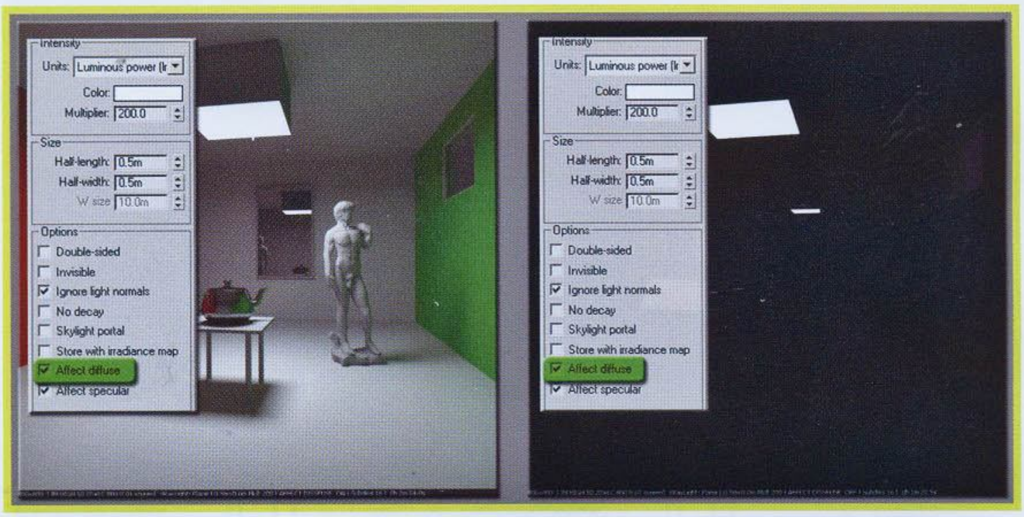
The GI is slower for the simple fact that *VRay* must compute the samples of shadows produced by ***VRayLights*** as well as the ones on geometries. It may seem like a waste of time, but this is by no means true, as during the final phase, rendering is considerably faster. It is enough to interpolate samples obtained during the ***IM*** computation. One can deduce that, with the option Store with irradiance map active, the better the ***IM*** quality, the better shadow quality. There is no doubt that the best system for shadow generation is always with ***Store with irradiance map*** deactivated. It all depends on the quality/time ratio which one wishes to obtain.
By observing the previous figure and the sum of rendering times, in this case the optimal solution is the one with ***Store with irradiance map*** deactivated. This is true up to a certain point, but what if one imagines wanting to fulfill a ***Fly-through*** animation? Frame times would increase dramatically and in this case the use of the option ***Store with irradiance map*** could certainly help save time, at the expense of a scarcely significant loss of quality. Another situation where this parameter is useful is when the use of a large number of ***VRayLights*** is necessary. In this case, the greater GI calculation time is compensated by the low time of the final render. Obviously the precision of shadows is directly connected to ***IM*** quality.
Observe the following image, created by enlarging a portion of the previous test.



■ **Figure 6.228**
Enlargement of a complex zone. Notice how the shadows generated with the ***Store with irradiance map*** option activated are smoother.

The leg of the table produces multiple shadows due to the presence of nine ***VRayLights***. Notice how, with the ***Store with irradiance map*** option, the shadow indicated by the arrow disappears. This is due to the insufficient quality of the ***IM*** and the excessive interpolation of samples. A greater ***IM*** quality, a greater number of Subdivisions and a lower interpolation of the ***IM*** could have solved this inconvenience, but one would have had an increase in GI calculation.

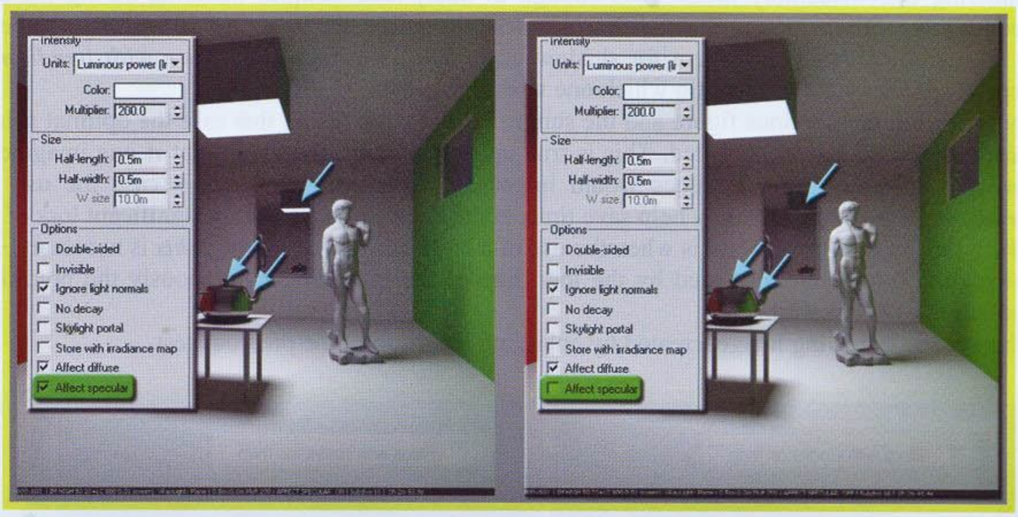
Affect diffuse - when this option is inactive, the ***VRayLight*** does not influence the diffuse channel of materials. The color or textures of the material are not illuminated and therefore, in this case, completely black.



■ **Figure 6.229**
With the ***Affect diffuse*** option, light does not influence the color of objects.

In this scene reflections are visible, and can be seen on the mirror and partly on the teapot.

Affect specular - this option allows one to deactivate the visibility of ***VRayLights*** on reflective objects.



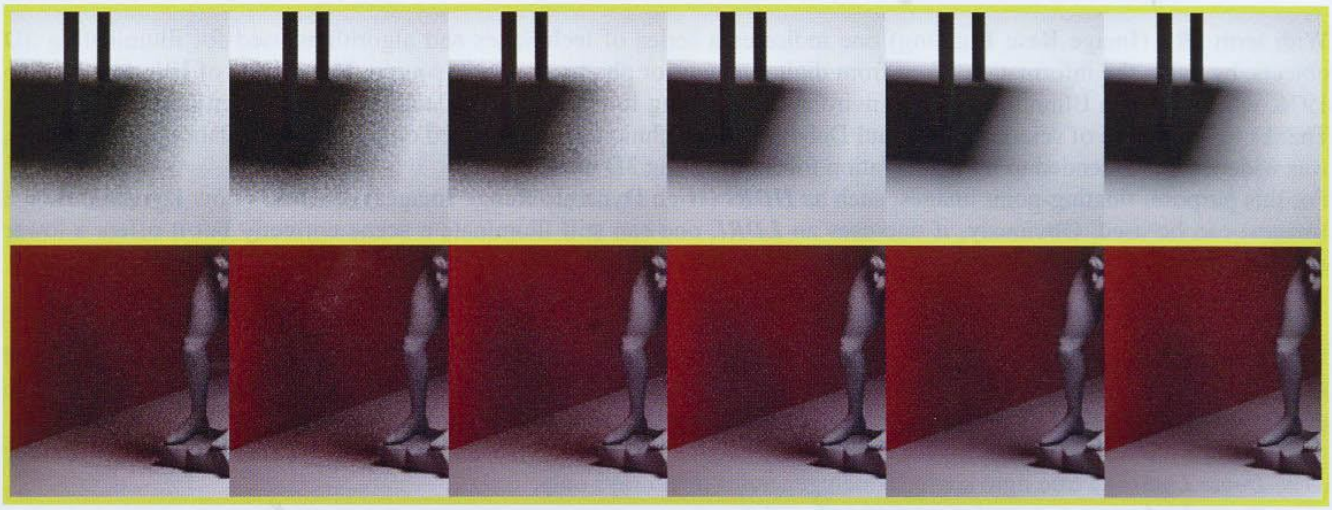
■ **Figure 6.230**
Deactivation of the ***Affect specular*** option.

Sampling

Subdivision - with this parameter it is possible to define the number of samples used by ***VRay*** for the computation of shadows generated by ***VRayLights***. Like in all parameters where ***Subdivs*** appear, the smaller its value, the briefer the rendering process, at the expense of quality and noise quantity.

■ **Figure 6.231**
Increasing the ***Subdivision*** value in ***VRayLights*** causes an increase in shadow quality and also in rendering time.

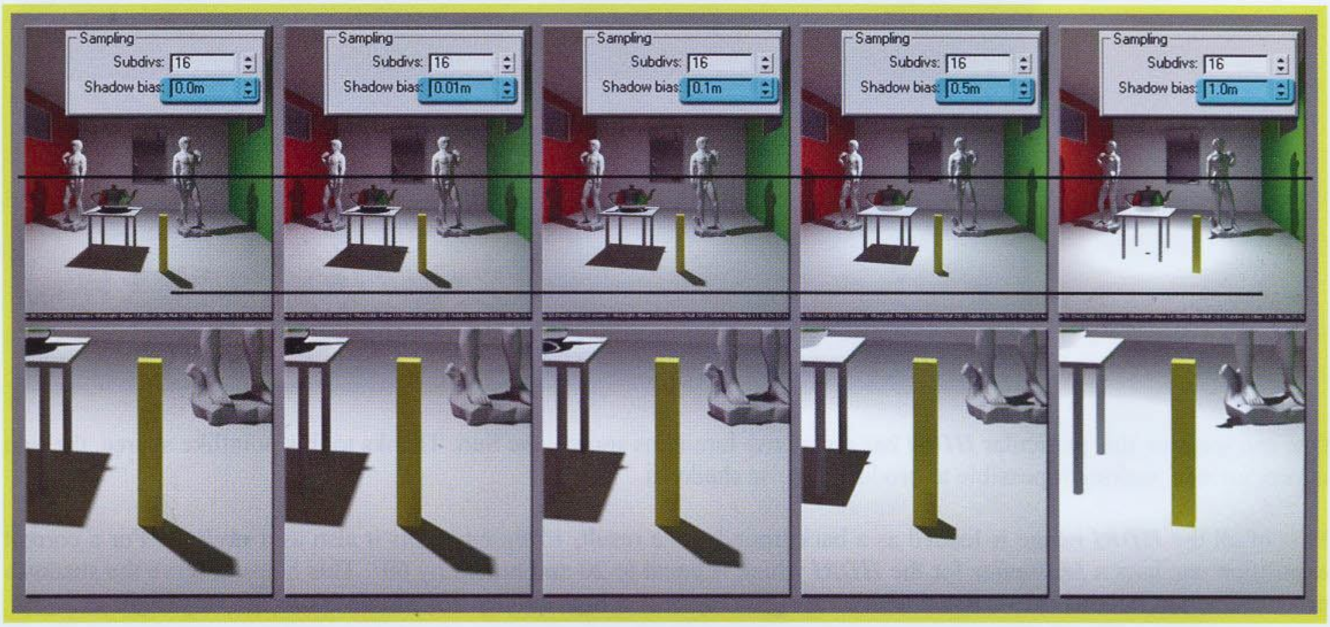
By using a very wide ***VRayLight***, ***VRay*** generates a shadow of the area type which is very smooth both under the table and on the two colored walls.



■ **Figure 6.232**
Detail of the ***VRayShadows***.

If one takes a closer look at the shadows produced, with values between 16 and 24 ***Subdivisions*** it has been possible to obtain a good quality shadow, within acceptable rendering times. Beyond 24 ***Subdivs***, in this particular scene there is no improvement in terms of noise reduction. Also, rendering time increases by almost 30%. One must bear in mind that the greater the surface of ***VRayLight***, the more ***Subdivisions*** must be used for a quality render.

Shadow bias - this parameter, also located in the ***VRayShadows params*** rollout, allows one to control the "depth" of shadows. The higher this parameter is set, the closer the shadow is to the geometry, until it disappears.



■ **Figure 6.233**
Effects due to the increase in ***Shadow bias***.

Dome light options

With term ***IBL*** (**I**mage **B**ase **L**ighting) one indicates a series of techniques and algorithms used for illuminating 3D objects, by using the information taken from digital images or photos as a light source. The origins of IBL date back to 1976, when a certain Blinn introduced Environment mapping for simulating reflections on computer generated objects. Thanks to the efforts of scientists like Paul Debevec, this technique, initially used only for the simulation of reflections, was widened and extended to light generation for illuminating 3D models.
For this purpose, floating-point images, such as ***HDRI*** (**H**igh **D**ynamic **R**ange **I**mages) or ***LDRI*** (**L**ow **D**ynamic **R**ange **I**mages) can be used. Obviously, if one uses an ***LDRI***, one can still illuminate a scene correctly but it offers a lower dynamic range.

VRay allows one to use ***HDRI*** images for Global Illumination. Thanks to ***Dome VRayLights***, one can use ***HDRI*** and ***LDRI*** without the help of Global Illumination. The two techniques, ***GI + Environment HDRI*** and ***Dome + HDRI*** can therefore seem similar, but there are important differences which make the ***Dome Light*** unique.

. The ***Dome Light*** is a true light source which generates direct rays only.
. The ***Dome Light*** does not produce stains, only noise.
. The ***Dome Light*** uses "***importance sampling***" technology. Thanks to this property, by using any image, one can obtain caustic effects and excellent shadows, without the use of high sampling levels, like for example using ***GI+HDRI*** with the ***IM+LC*** or ***IM+QMC*** system. In fact *VRay* automatically analyzes the image, by sampling the areas with a greater luminosity. It is an adaptive technique and so a greater number of rays is released from the brighter areas compared to the darker ones.
This allows sharper and more precise shadows to be produced compared to the results one would get with the ***GI+HDRI Environment*** system.

A simple outdoor scene will be rendered, using a 1024x512 pixel ***HDRI*** map.

■ Figure 6.234
Different exposure levels of the *HDRI* being examined.

One can see how this particular ***HDRI*** has an intense luminous source, the Sun. Thanks to this pointlike source, the Sun is very intense, making it possible to produce precise shadows.

First of all the ***HDRI*** image is loaded as a background. As a result, *VRay* interprets it also as a ***skylight***. For a correct exposition one uses a ***Multiplier*** for the ***HDRI*** which is equal to 20 and turned by 60°. This helps observe the shadows produced better. All this is calculated with a ***IM+QMC*** system at high quality. ***IM Subdivisions*** are very high.

■ **Figure 6.235**
HDRI used for generating ***skylight***.

After 30 minutes of rendering, the result is still not satisfactory. In order to capture as many details as possible, especially in shadowy areas, one must resort to high settings of ***Min rate*** and ***Max rate*** with 150 ***Subdivisions***. By using interpolations which are higher than samples, the situation does not improve. The gain in terms of cleanliness of the picture is evident, but shadows tend to blend to much and lose definition.

■ **Figure 6.236**
Increase in interpolation for the ***Irradiance Map***.

Observing the use of the ***HDRI*** as ***skylight***, we will now analyze its behavior as ***Dome Light***.

First of all one must deactivate the GI and eliminate the 3ds Max background. Create a ***Dome Light*** and use an ***HDRI*** as a texture. Having the invisible option deactivated by default, the ***HDRI*** is used automatically by the ***Dome Light*** as a Background image. By activating the invisible option, the background becomes black, as indicated by the 3ds Max Environment. Lastly the ***Affect Specular*** option is active and as a result the ***HDRI*** is used as an environmental map for reflections. Also in this case, its deactivation lets the ***VRay Environment*** manage reflections.

■ **Figure 6.237**
Activation of the ***Dome Light*** with use of a ***HDRI***.

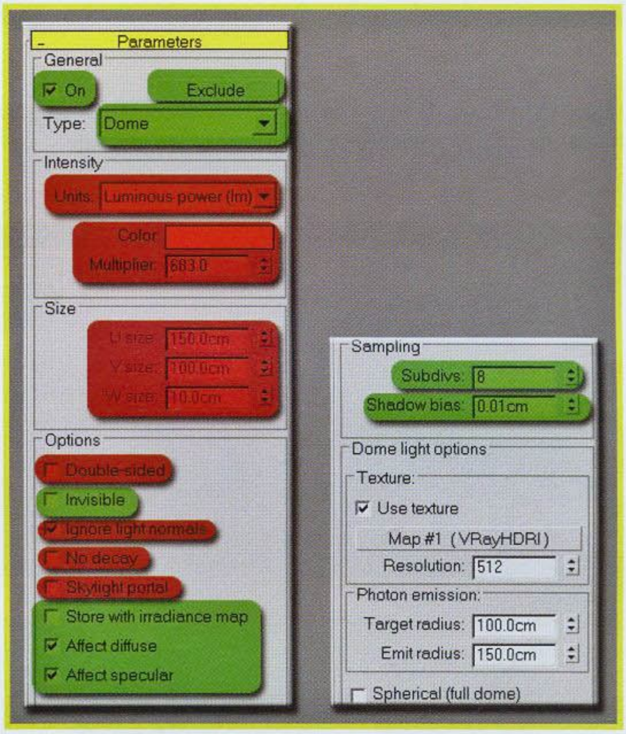
The result is similar to the previous one but there are substantial differences. The first is that statues are darker compared to the previous images. This is because there is no indirect illumination and the only light source, the ***Dome Light***, produces only direct light. The second is the quality of shadows, definitely better than the previous ones. Third, the type of render, very low. The next step consists in activating the GI, still with ***IM+QMC***. This time, standard values will be used: ***IM*** with ***medium preset***, ***Subdivs*** = 50, ***Interp. Samples*** = 20, ***QMC Subdivs*** = 8 and ***Secondary bounces*** = 3.

■ **Figure 6.238**
Activation of the ***Dome Light*** with use of an ***HDRI*** and Global Illumination.

The images speak for themselves. The GI is very clean, rendering time is very low and the quality of shadows generated by a texture and not by a "light" are very high quality. Amazing!

■ **Figure 6.239**
Comparison between different renders. There is little to be said, the last one is definitely the best!

By using a texture as a light source within a ***VRay Dome Light***, one notices that not all the options found in the ***Parameters*** rollout which have been analyzed in the previous pages are useful and above all, active.



■ **Figure 6.240**
The parameters which do not affect the result of a render when using ***Dome Light*** are highlighted in red. The ones which instead influence the result are in green.

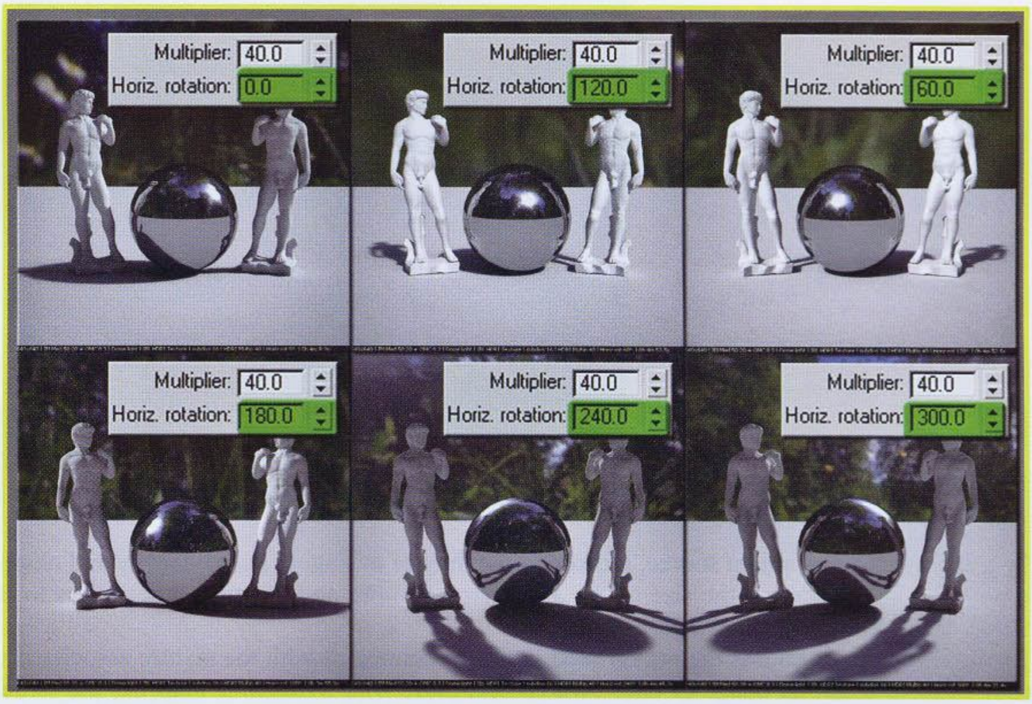
By using a texture as a light source, it takes on the duty of managing light intensity and obviously color intensity. There is no point in considering the size of the ***VRay Dome Light***, and likewise it is useless to consider using the ***Double-Sided***, decay or ***Skylight portal*** parameters.
The invisible option causes the background to not be connected with the texture associated with the ***Dome Light***. This allows one to insert an image of ones own choice. ***Store with irradiance map*** has the same function as always, as do ***Affect Diffuse***, ***Affect specular***, ***Subdivs*** and ***Shadow bias***.

A note regarding intensity and direction of the ***HDRI***. As we have already described in the chapter regarding the ***VRayHDRI*** map, ***VRay*** allows one to control its ***Multiplier***, both for horizontal and vertical rotation, as well as other parameters. It is therefore possible to manage both the intensity of the ***Dome Light*** and the direction of shadows by intervening directly on the ***HDRI*** map within the Material Editor. By increasing the ***Multiplier*** of the ***HDRI*** ones makes the map more luminous and therefore the emission of luminous flux is more intense. As a result, the scene is more illuminated. Similarly to what happens in situations with a lot of light, rendering times increase in proportion to the quantity of light in the scene.

■ **Figure 6.241**
Variation of intensity in the ***HDRI*** map used within the ***VRay Dome Light***.

In order to modify the position of shadows, it is enough to rotate the ***HDRI*** map. In the following image the horizontal rotation has been changed, but it is equally possible to modify the vertical one.

■ **Figure 6.242**
Variation of the rotation of the ***HDRI*** map. It is in any case possible to act upon the remaining parameters of the ***HDRI*** map, such as ***Flip horizontally*** or ***Vert. Rotation***, producing changes in the scene's illumination.



Another method for changing the position of shadows is to physically rotate the ***Dome Light*** in the viewport. As seen in the paragraph about the ***Dome Light***, it has been showed how rotating the X or Y axis brought changes to illumination. In this case we have the same situation. By changing the X or Y rotation of the ***Dome Light*** one obtains different illumination effects, because one is rotating the "invisible cap" created with the ***Dome Light***. The rotation along the Z axis does not cause any changes instead.

Use texture - when one selects the ***VRayLight Dome***, ***VRay*** offers a series of parameters which are apt for controlling the internal ***IBL*** system. With this option the possibility of inserting a texture is activated, only an ***HDRI***, to be used as a light source. By deactivating the ***Use texture*** option, the ***Multiplier*** and the color of the ***Dome Light*** depend on the parameters of the ***VRayLight*** and no longer on the ***HDRI*** map.

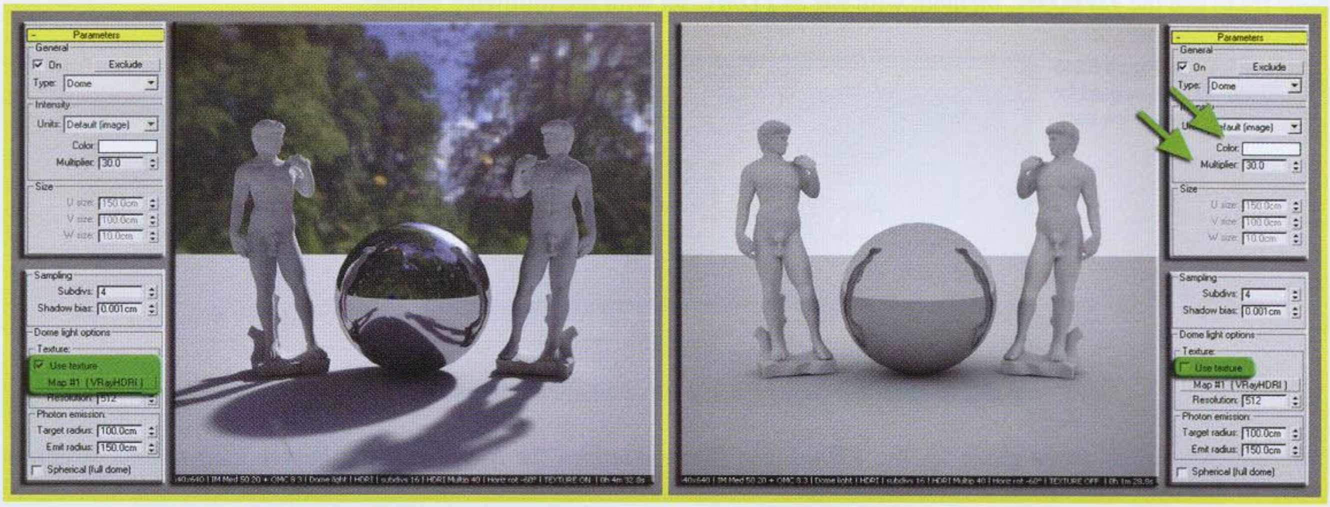


Figure 6.243
Deactivation of the ***HDRI*** texture applied to the ***Dome Light***.

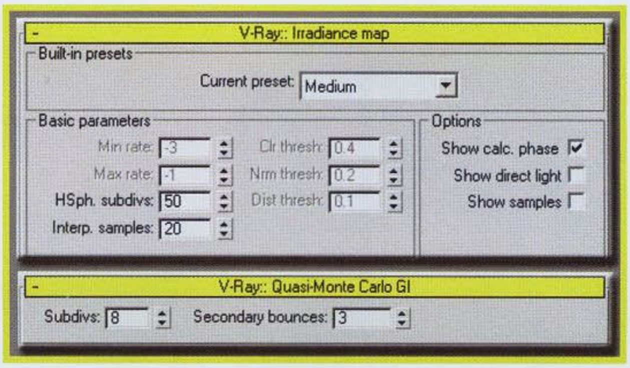
Resolution - this specifies the resolution which the ***HDRI*** map will be resampled at for ***importance sampling***. Remember, ***importance sampling*** exploits the fact that each pixel of the ***HDRI*** has an amount of "energy". The more energy these pixels have, the more likely it is that they will be used for illuminating the scene. It is therefore obvious that the choice will fall on the pixels which contain most energy. In the previous case, ***importance sampling*** will have sampled mainly the pixels contained within the sun.

In order to understand the way this parameter works, observe the following images. The ***HDRI*** map used so far has a resolution of 1024x512 pixels. First of all the rendering process will take place with the ***LC+LC*** method, with setting of ***Subdivs*** = 600, ***Size Screen*** = 0.01 and filtering disabled. This allows one to observe the effects of the change in resolution much better.

Figure 6.244
Effects of the ***Resolution*** parameter on the GI quality.

There are two effects due to the increase in sampling resolution. The first one concerns rendering time. With low resolution renders are faster, but because the resolution of sampling is so low and having used so few samples, the ***LC*** does not have enough information to generate an accurate map. Already with a resolution of 64 pixels the situation improves greatly, whereas between 512 and 1024 pixels there is not much difference.

Now try to use the ***IM+QMC*** method, with ***IM*** and ***QMC*** set with the following values:



■ **Figure 6.245**
The ***IM*** and ***QMC*** values used for the next few tests.

The only adjustment to be made for the ***VRayLight Dome*** involves the ***Store with irradiance map*** option, which is active in this case. The results are the following.

■ **Figure 6.246**
Effects of the ***Resolution*** parameter on the GI, by using the ***IM+QMC*** method.

In this case the difference is less evident compared to the ***LC+LC*** method. The lower quality of the GI when using low resolutions is visible in any case. The reason is that the samples of the ***IM*** do not have sufficient information for creating a quality map and because of this the result is a stained ***IM***. If one observes the ***IM*** generated by each test one can confirm that samples are more or less distributed in the same way. From this we can deduce that the scarce quality of the render is due to the low number of samples generated by the ***Dome Light*** and their low quality.

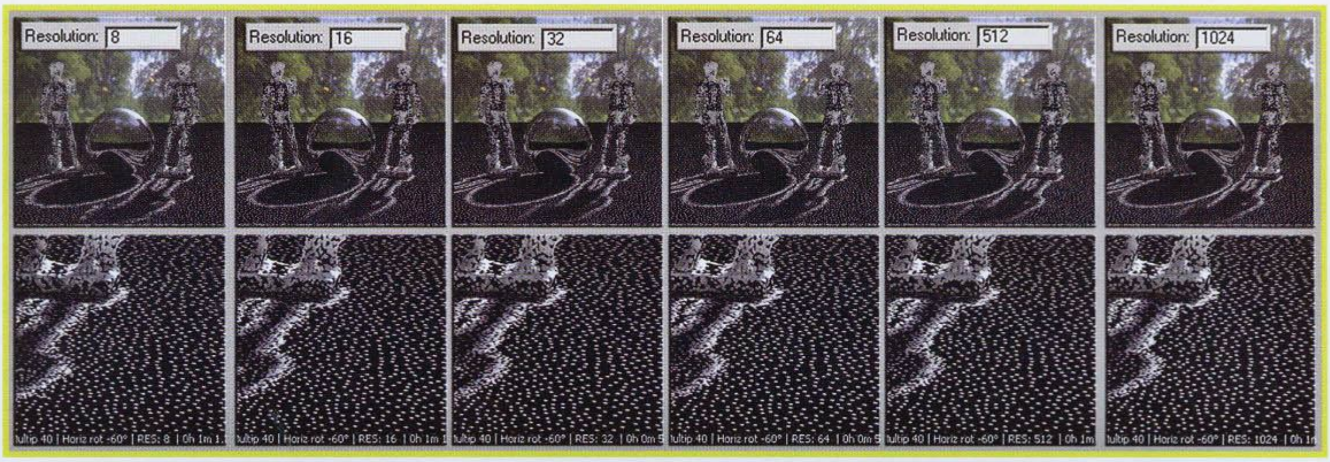


Figure 6.247
Irradiance map at different resolutions.

To finish, a series of tests fulfilled with ***HDRI*** maps at different resolutions. Not only at 1024x512 pixels, but at 64x32 pixels and 512x256 pixels. In this case, the parameter ***Resolution*** in the ***VRayLight Dome*** has been maintained as corresponding to the resolution of the ***HDRI*** texture used in that moment.

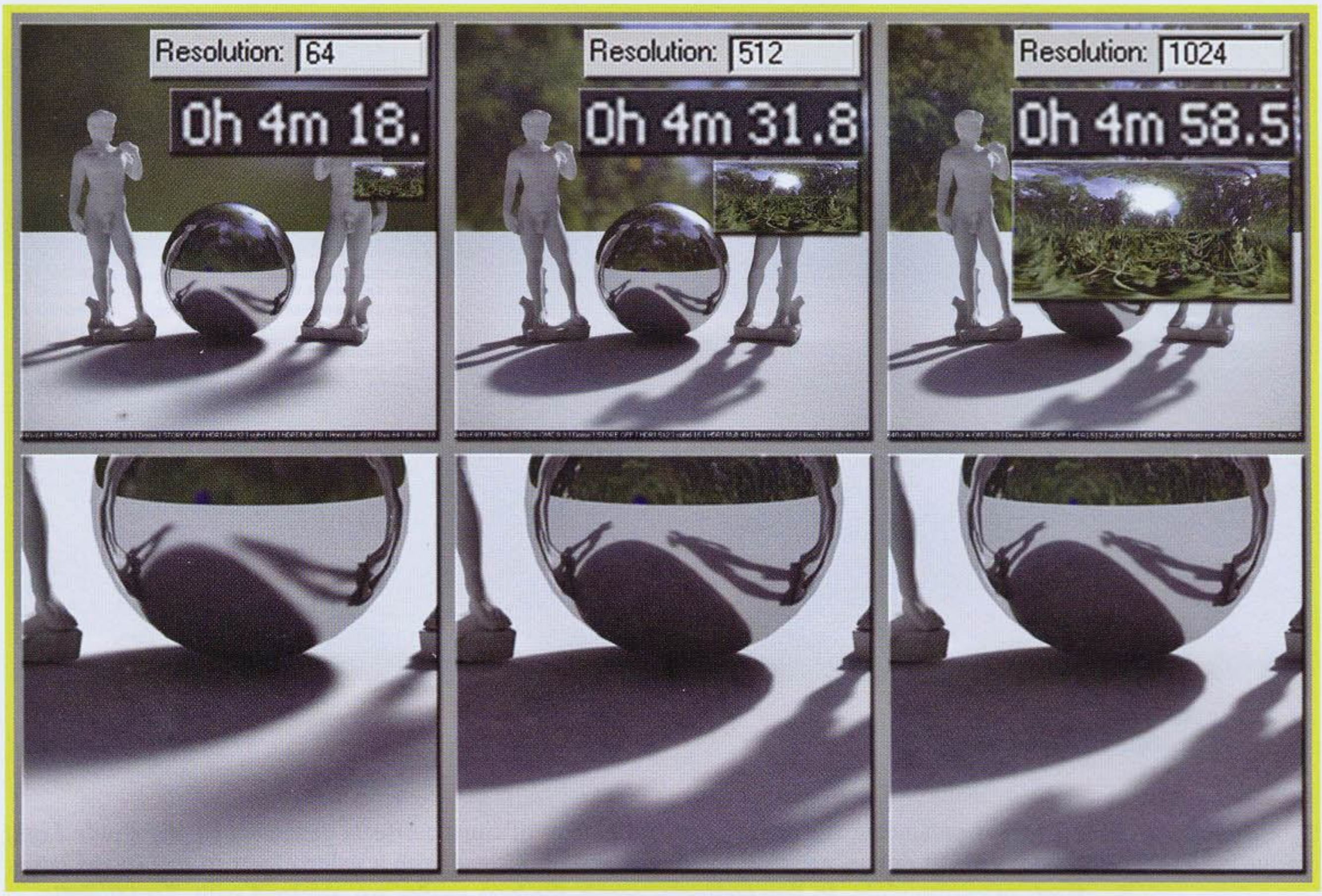


Figure 6.248
Use of ***HDRI*** maps at different resolutions within the ***VRayLight Dome***.

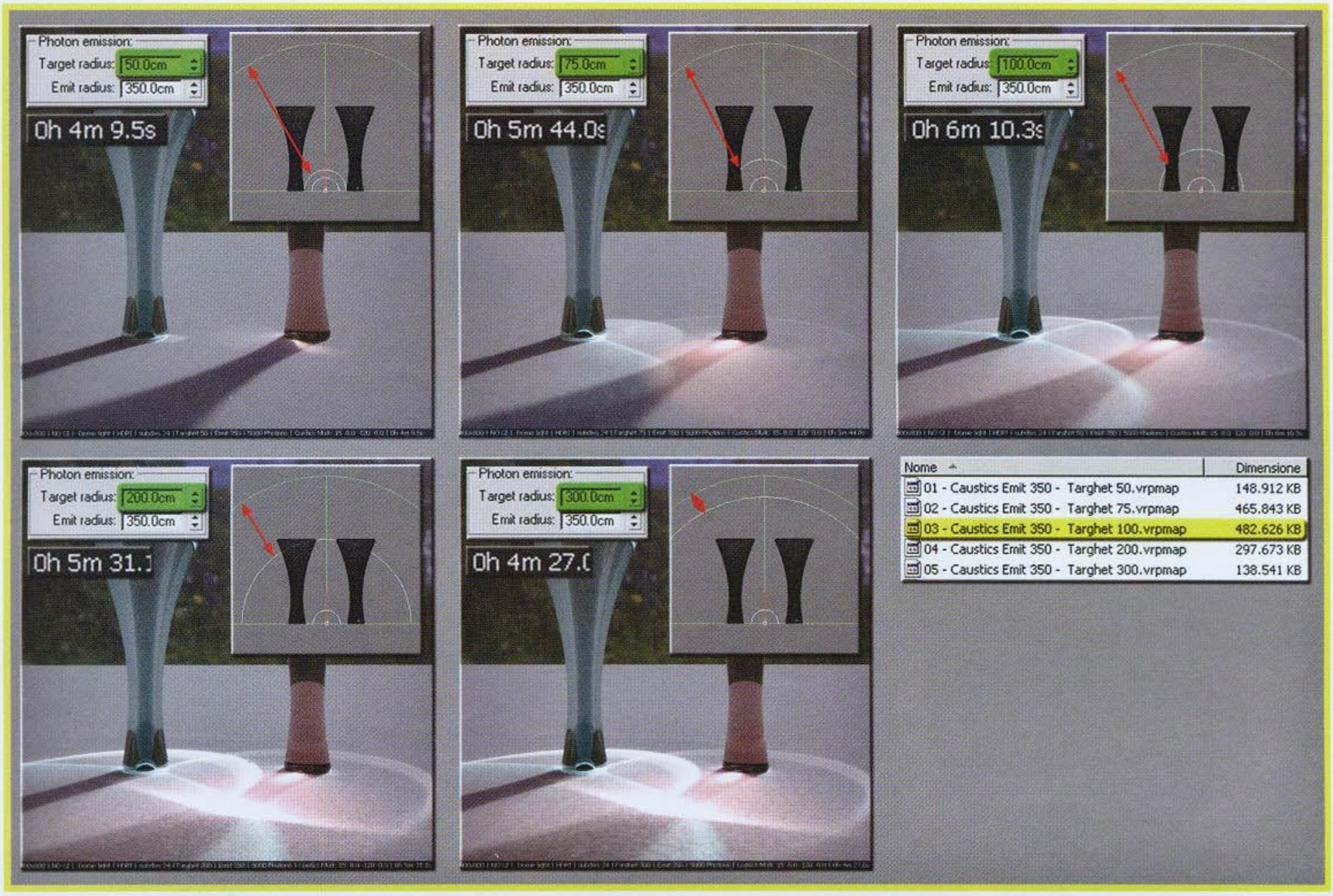
By deactivating the ***Store with irradiance map*** option and therefore using an unbiased system for the calculation of shadows, the only difference perceivable between renders is the quality of shadows, less precise at very low resolutions.

Target radius - this parameter allows one to define the size of the semisphere around the ***Dome Light*** from which photons are emitted for the generation of caustic effects. It is the internal semisphere.

Emit radius - this defines the size of the semisphere around the ***Dome Light*** from which photons are emitted towards the smaller sphere defined by the parameter ***Target radius***. It is the outer semisphere.

By observing the ***Dome Light*** in viewport, we realize that it is enveloped within two semispheres. A smaller one with a radius controlled by the ***Target radius*** parameter and a larger one with its radius controlled by ***Emit radius***. This last parameter, being the one which emits photons for the calculation of the ***Photon Map*** towards the internal sphere, can never be smaller than ***Target radius***. *VRay* itself does not allow this to happen. In fact, if we try to lower ***Emit radius*** below the size of the ***Target radius***, *VRay* automatically starts lowering also the level of ***Target radius*** when the value of ***Emit radius*** draws very close. The two spherical caps are in fact limited so that they can not even have the same value but the size of ***Emit radius*** is always at least slightly greater.

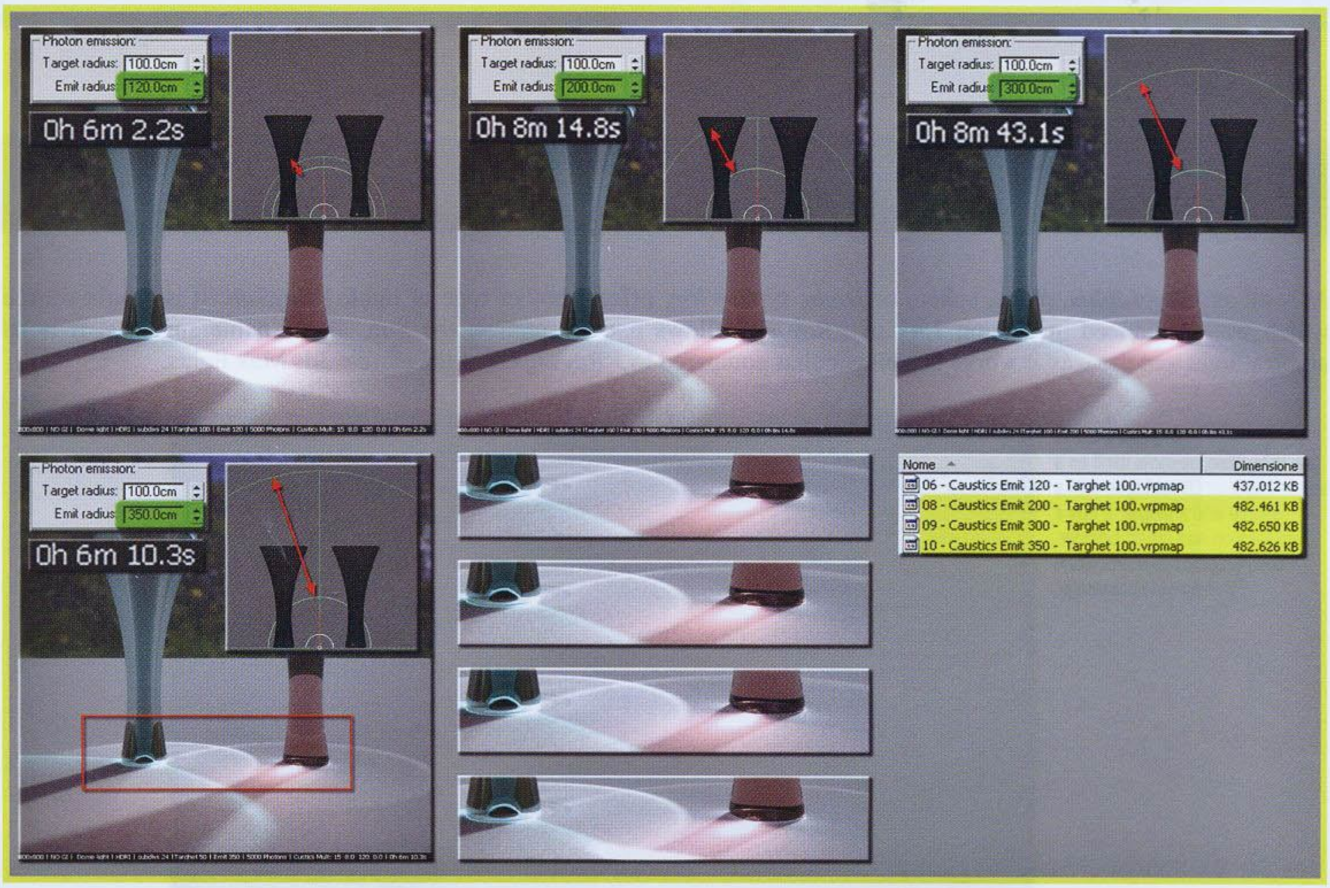
To understand these two parameters observe the following image. There are two glass pillars placed 200 cm away from each other. There is a ***Dome Light*** between the two geometries. The outer radius, ***Emit radius***, englobes the pillars completely. Without activating the GI, a series of images will be calculated by changing only the ***Target radius***. For the generation of caustic effects, the relative parameter within the ***VRay: Caustics*** rollout has been activated. Also, to define the number of emitted photons, one has intervened upon the properties of the ***Dome Light***. In any case, for more information concerning caustics, it is best to read the paragraph concerning that subject.



■ **Figure 6.249**
Variation of the ***Target radius*** parameter.

By changing the ***Target radius*** parameter, the appearance and quality of caustic effects generated by the ***Dome Light*** change radically. With a very small radius, caustics are almost absent, instead with a value closer to that of the larger semisphere, caustic effects are quite vivid. In this particular scene, the best results are obtained with ***Target radius*** = 100 and ***Emit radius*** = 350, that is, 1/3rd of the greater semisphere. By looking at the generated ***Photon Maps***, one can see that the ***Dome Light*** set to 100 is the one which weighs more in terms of Bytes. This means that it contains more information than the other situations.

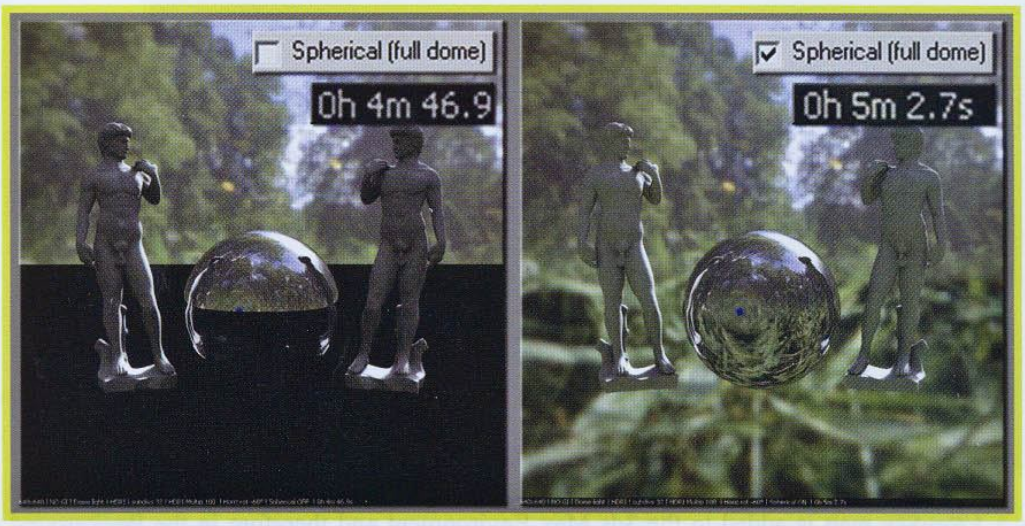
In the following images, on the contrary, ***Emit radius*** is modified, leaving the ***Target radius*** unaltered, set to 100.



■ **Figure 6.250**
Variation of the ***Emit radius*** parameter.

Both from the images and from the size of the ***Photon Map***, one can tell that values above 200 do not change anything. In order to obtain caustic effects from an ***HDRI*** map it is necessary to find the right compromise between ***Target*** and ***Emit radius***.

Spherical - by activating the ***Spherical*** option, the ***Dome Light*** generates a Spherical ***Environment*** instead of the usual semisphere.

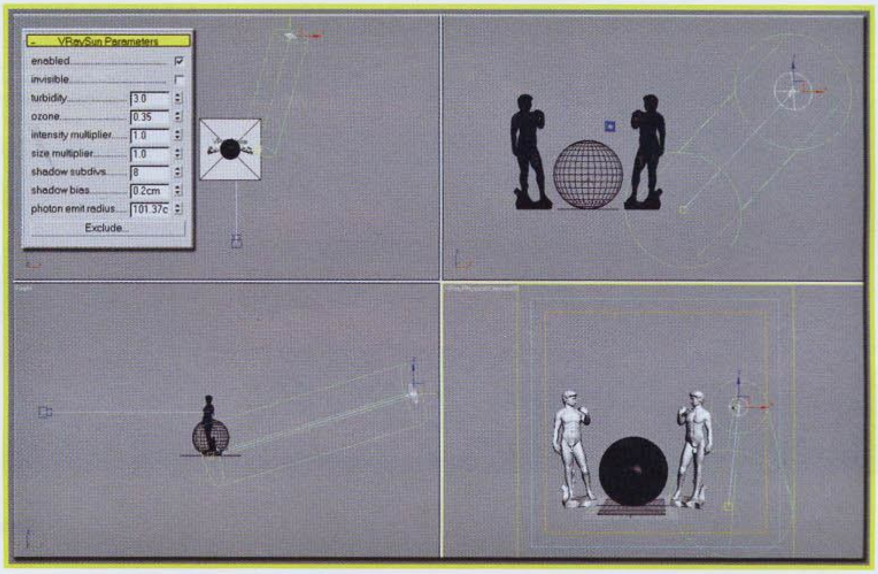


■ **Figure 6.251**
By activating the ***Spherical*** parameter, the ***Dome Light*** uses a complete sphere instead of the semisphere.

VRaySun and VRaySky

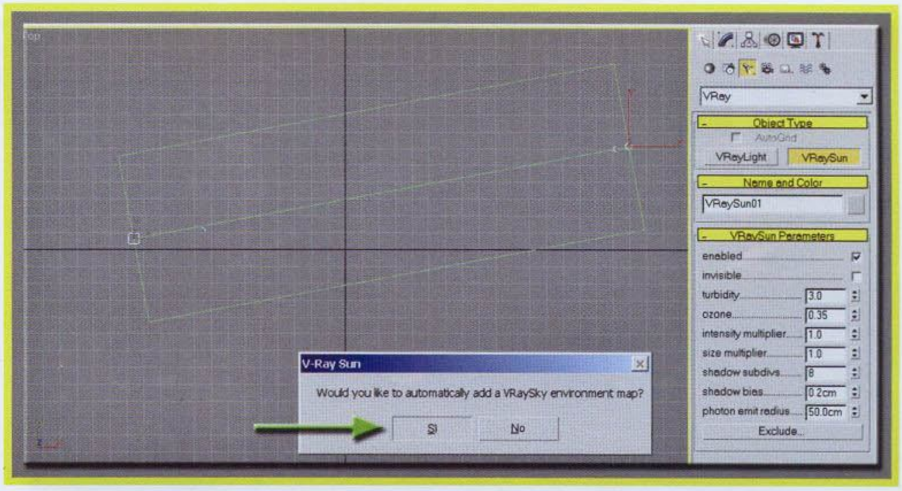
INTRODUCTION

As well as ***VRayLights***, plane, sphere or ***Dome*** type, ***VRay*** offers another type of light, ***VRaySun***. It is a light source which was introduced in version 1.47 of ***VRay*** and allows one to simulate the Sun, as the name suggests. It is an atypical light and the way it works is a bit peculiar compared to ***VRayLights*** or the classic 3ds Max lights.



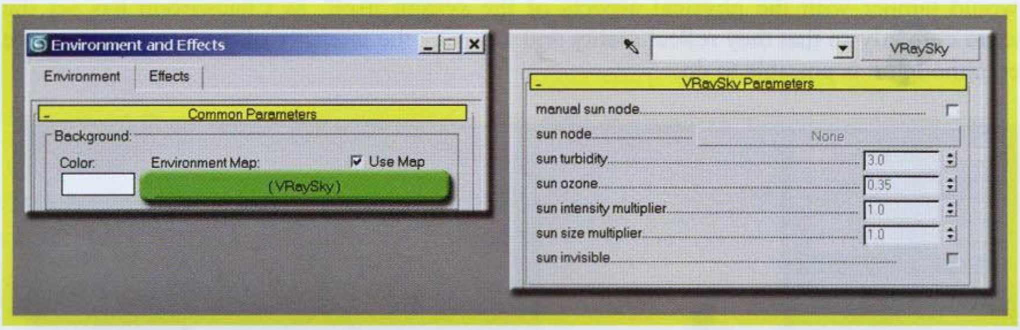
■ **Figure 6.252**
The ***VRaySun*** presents itself as a sort of 3ds Max Direct Light, with a source and a ***target***. It has a circular icon and green cylinder. These graphic representations have precise meanings.

In actual fact the creation of a physical system for simulating the sky requires not only a light source, the ***VRaySun*** in this case. This light, which we could for now compare to a 3ds Max Direct Light and which represents the Sun, is not enough to represent physical Daylight within ***VRay***. There is also the need for something which represents the background or the sky. This role is given to the ***VRaySky*** map. These two types of objects, ***VRaySun*** and ***VRaySky***, work together, in symbiosis. In the v1.5 RC3, when one creates and drags a ***VRaySun light***, ***VRay*** automatically asks us if we wish to insert a ***VRaySky*** map into the 3ds Max Environment.



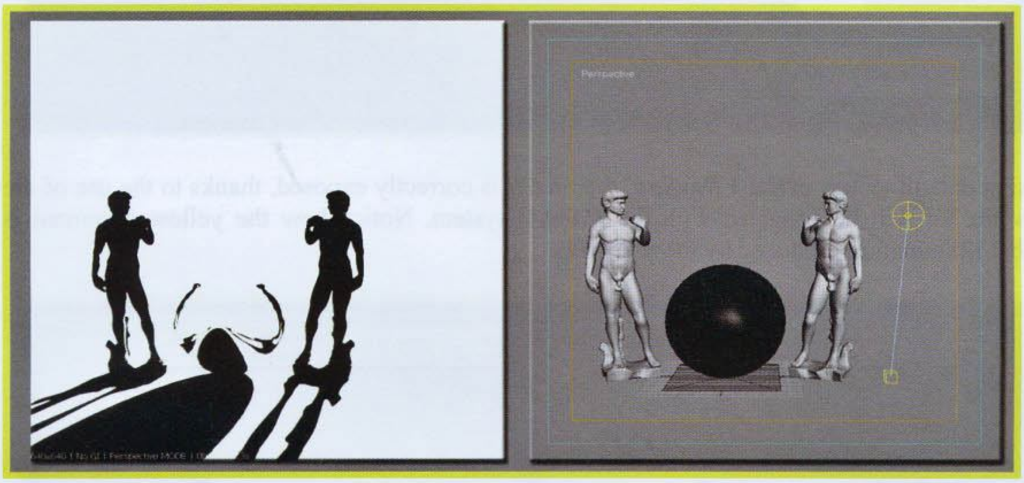
■ **Figure 6.253**
Automatic creation of ***VRaySky***.

If we confirm this question, a ***VRaySky*** map is added to the 3ds Max Environment and therefore in the Background. Once the Environment and Effect panel of 3ds Max has been opened, ***VRaySky*** appears in the Environment Map slot. At this stage, it is enough to drag it into any free slot of the Material Editor for following modifications.



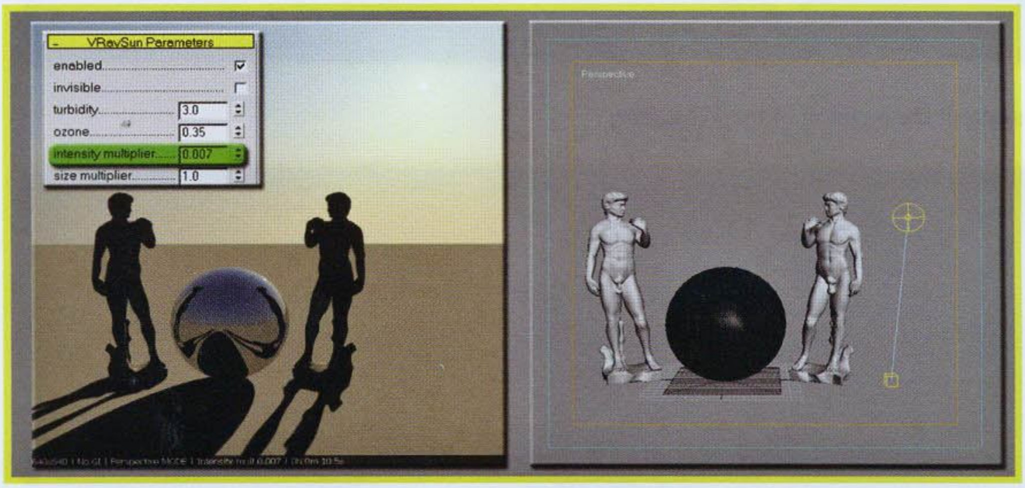
■ **Figure 6.254**
The ***VRaySky*** placed within the Environment.

Within the scene one can find both ***VRaySun*** with default values and ***VRaySky***. This is the result without activating the Global Illumination.



■ **Figure 6.255**
Render with a VRay solar system. The render was made by observing the scene with perspective view mode.

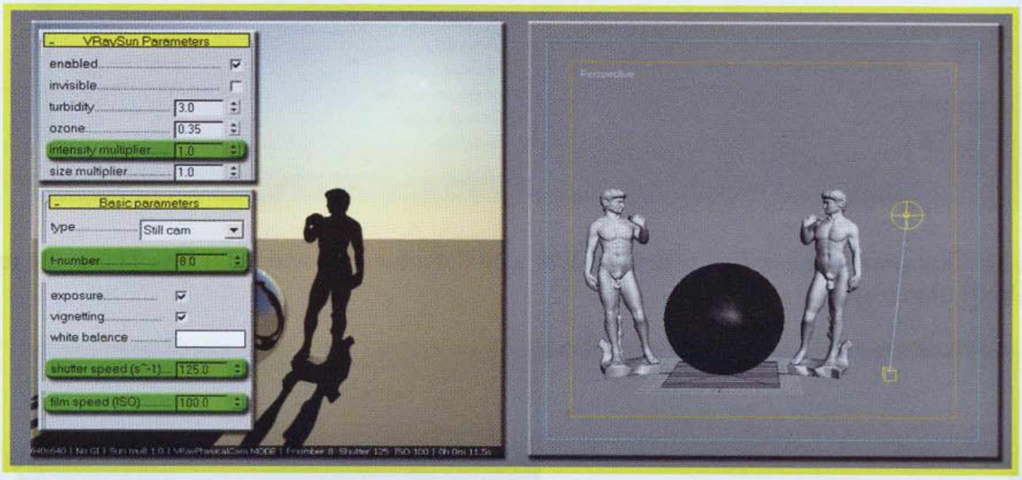
It cannot be said that this render is photorealistic! There is an explanation for this though. The method we have just used is an illumination system which attempts to recreate real situations found in nature. One can understand how the luminous flux of the Sun simulated in *VRay* is extremely intense, just like in the real world. A solution to bring the exposure of the render back to normal levels could be for example that of manually lowering the intensity of the ***VRaySun***. In fact the result is now much better.



■ **Figure 6.256**
The ***VRaySun*** with the ***intensity multiplier*** lowered. Now the render is no longer overexposed. Notice the light brown color of the plane, due to the yellow dominant of the Sun. In truth the materials are all gray.

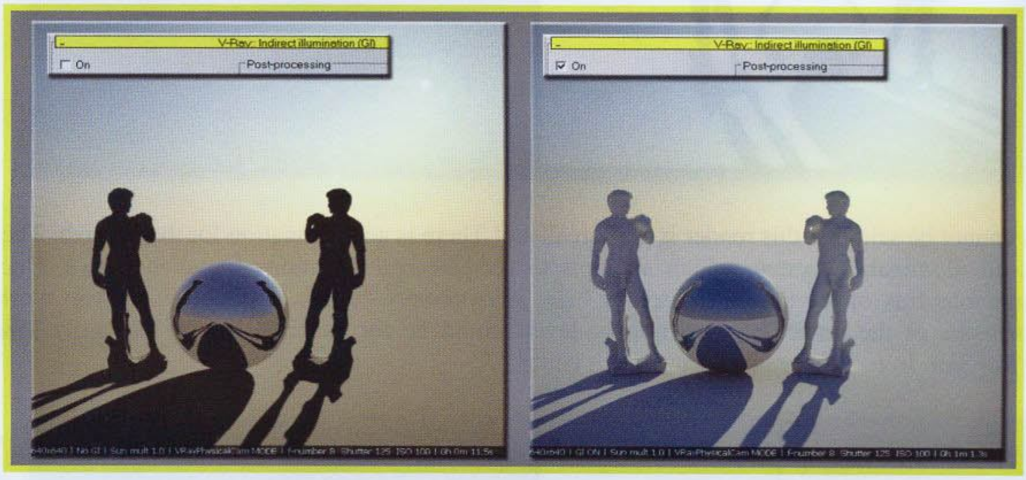
By lowering the ***multiplier*** of the ***VRaySun*** not only the intensity of the Sun decreases, which can be seen by the fact that now the plane is visible, but also the background, defined by a ***VRaySky*** map, is correctly exposed. From this we can tell that between ***VRaySun*** and ***VRaySky*** there is a connection: changes applied to ***VRaySun*** also affect ***VRaySky***. For what reason have ***Chaosgroup*** decided to set ***VRaySun*** with the ***multiplier*** = 1 if every time we must lower it? The system we have just used is physically correct. For this reason we must use a system of observation which is also physically accurate. It is necessary then, to use the ***VRayPhysicalCamera***, as one does when using ***VRayLights*** with physical units. Lowering the ***VRaySun multiplier*** to a value near 0.0 is a solution which it is best to avoid, if possible.

This last solution, in fact, would tamper with the physical aspects of the system itself, as Chaosgroup has set the parameters concerning ***VRaySun*** correctly so that they reflect reality and lowering the intensity of the Sun would not be a physically correct maneuver, although it is possible to do.



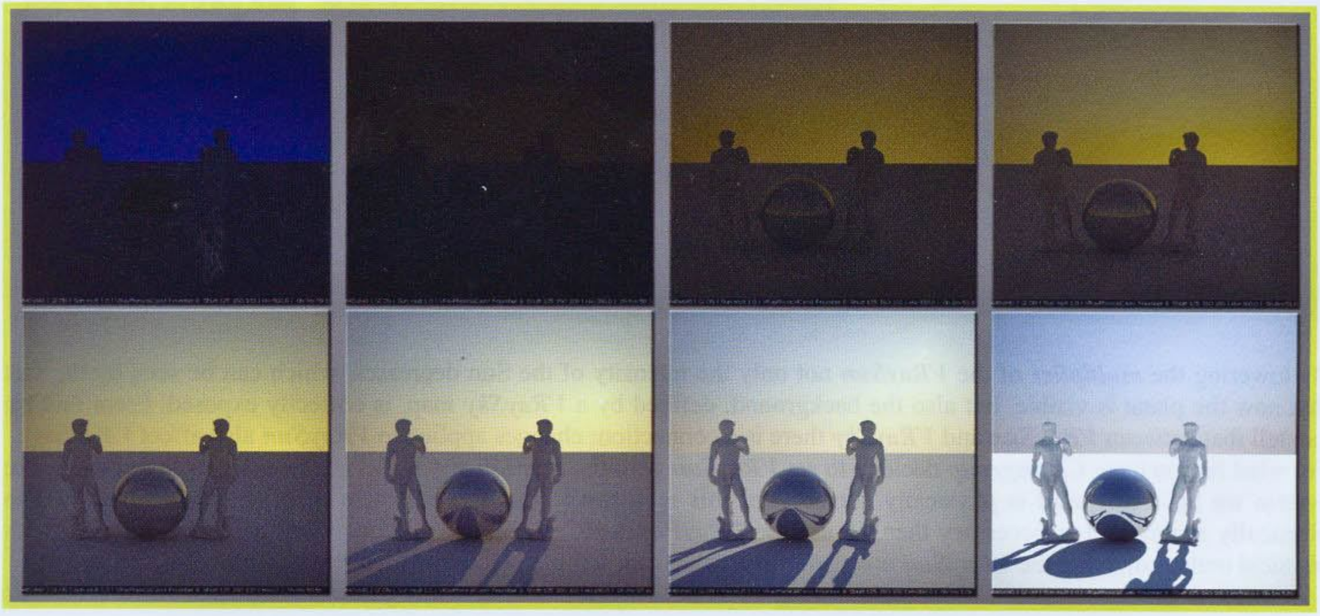
■ **Figure 6.257**
Use of the ***VRaySun+VRaySky*** together with the ***VRayPhysicalCamera***.

In this example, although using default values of the ***VRaySun***, the render is correctly exposed, thanks to the use of the ***VRayPhysicalCamera***. Now the GI will be activated with the ***IM+LC*** system. Notice how the yellow dominant is contrasted by light blue Global Illumination produced by the ***VRaySky***.



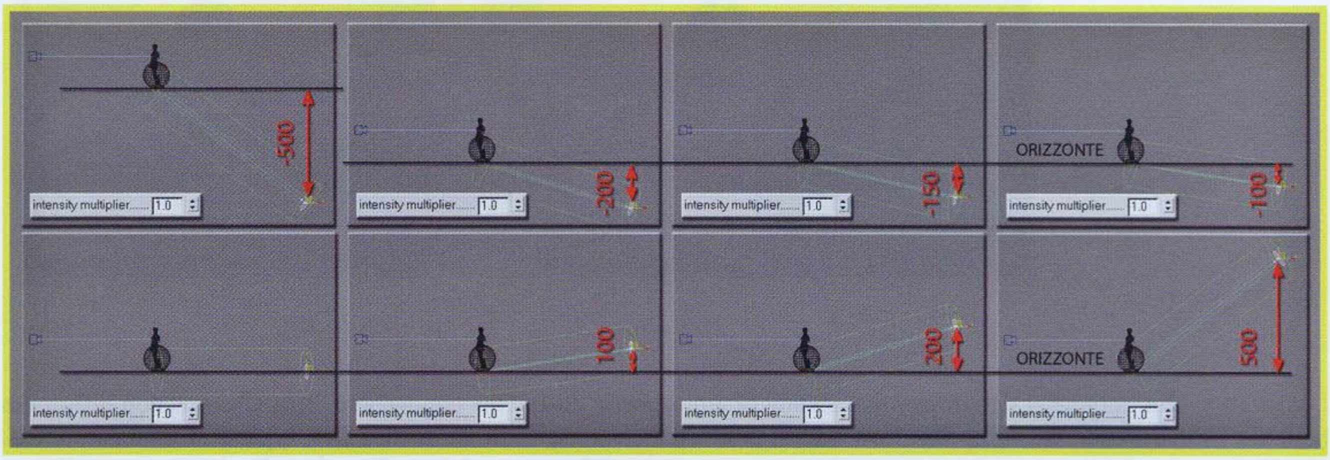
■ **Figure 6.258**
GI activation.

A last consideration to be analyzed: the position of the ***VRaySun***.



■ **Figure 6.259**
Variation of light conditions in the scene generated by changing the height of the ***VRaySun*** alone.

The change in height of the ***VRaySun*** influences ***VRaySun*** and ***VRaySky*** color/intensity directly. In the previous examples, during which only the position of the Sun has been modified, both the amount of light and the background map and therefore also the GI have been modified by the position of the ***VRaySun*** alone. By lifting the ***VRaySun*** along the vertical axis, *VRay* modifies the intensity of the Sun automatically. Notice how its ***intensity multiplier*** is always equal to 1. So how is this difference in illumination possible? This happens because this system has been conceived in such a way to produce a physically accurate behavior of the Sun. In truth the higher up the Sun is, the more the Earth is illuminated. Both in the morning and in the evening the Sun emits the same amount of light, so the ***intensity multiplier*** = 1. It is the visibility of the Sun, atmospheric conditions and the different height on the horizon which make the Sun illuminate less in certain moments of the day.

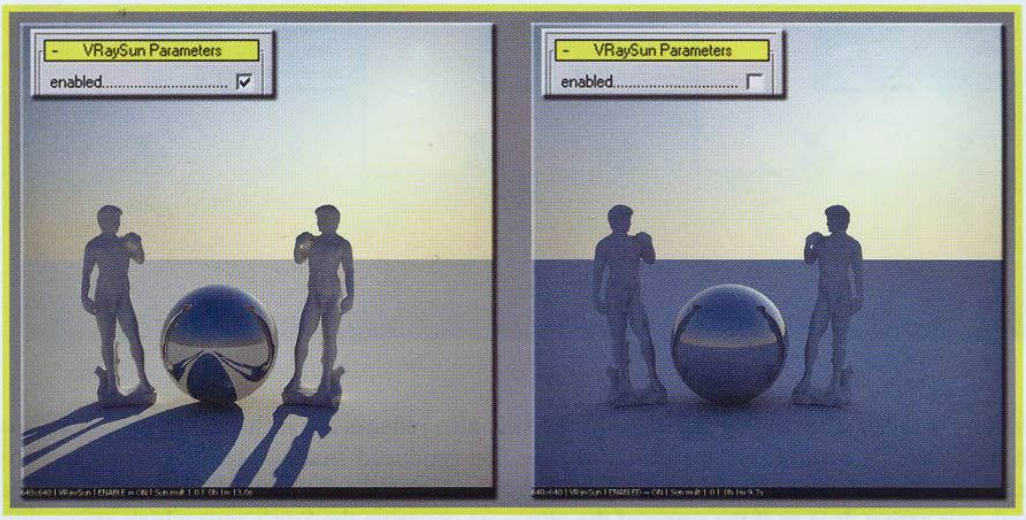


■ **Figure 6.260**
Variation in the height of the ***VRaySun***.

VRaySky is a parametric map which is changed according to the position of the ***VRaySun*** light source. The user must not change its parameters, because the ***VRaySun*** sees to this, without this being visible to the user. When the ***VRaySun*** is high up in the sky, ***VRaySky*** is, for example, light blue, typical of a Summer day in the hotter hours. Vice versa, when the Sun is on the horizon, the ***VRaySky*** map takes on an amber color, simulating the color of a sunset.

PARAMETERS VRaySun

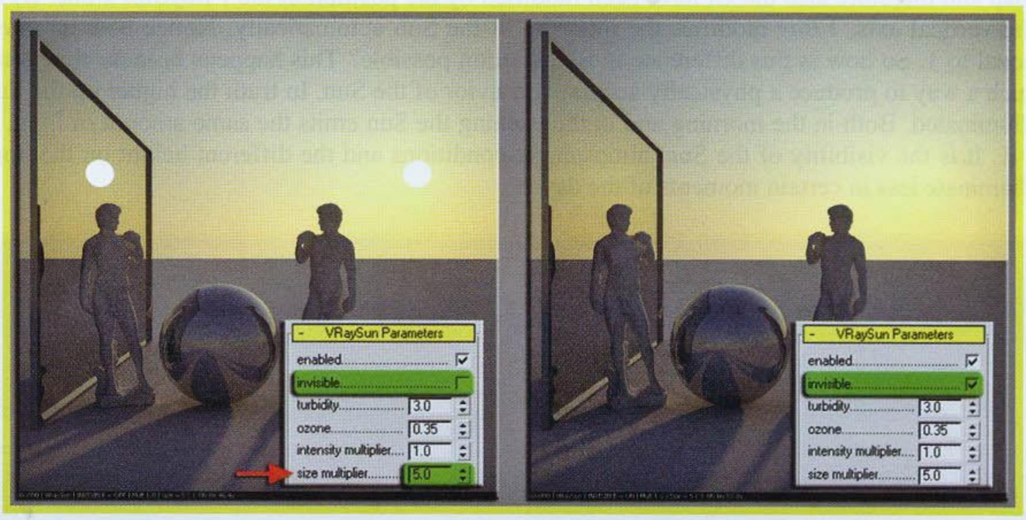
enable - this activates or deactivates the ***VRaySun***.



■ **Figure 6.261**
Deactivation of ***VRaySun***.

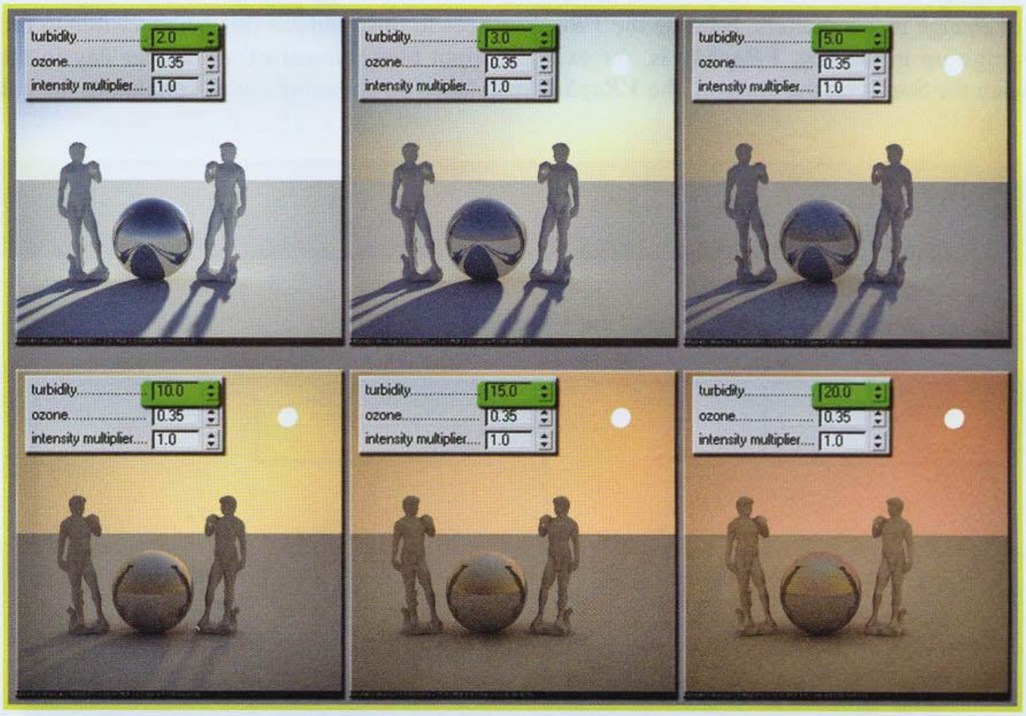
By deactivating the ***VRaySun***, the scene is illuminated only by the ***VRaySky*** map, which is in any case influenced by the position of the ***VRaySun***. The shadows caused by the ***VRaySun*** are no longer present, and neither is its typical yellowish color. In this case, the scene is illuminated by the ***skylight*** generated by the ***VRaySky*** map.

invisibile - this controls the visibility of the Sun both for reflections and for the camera.



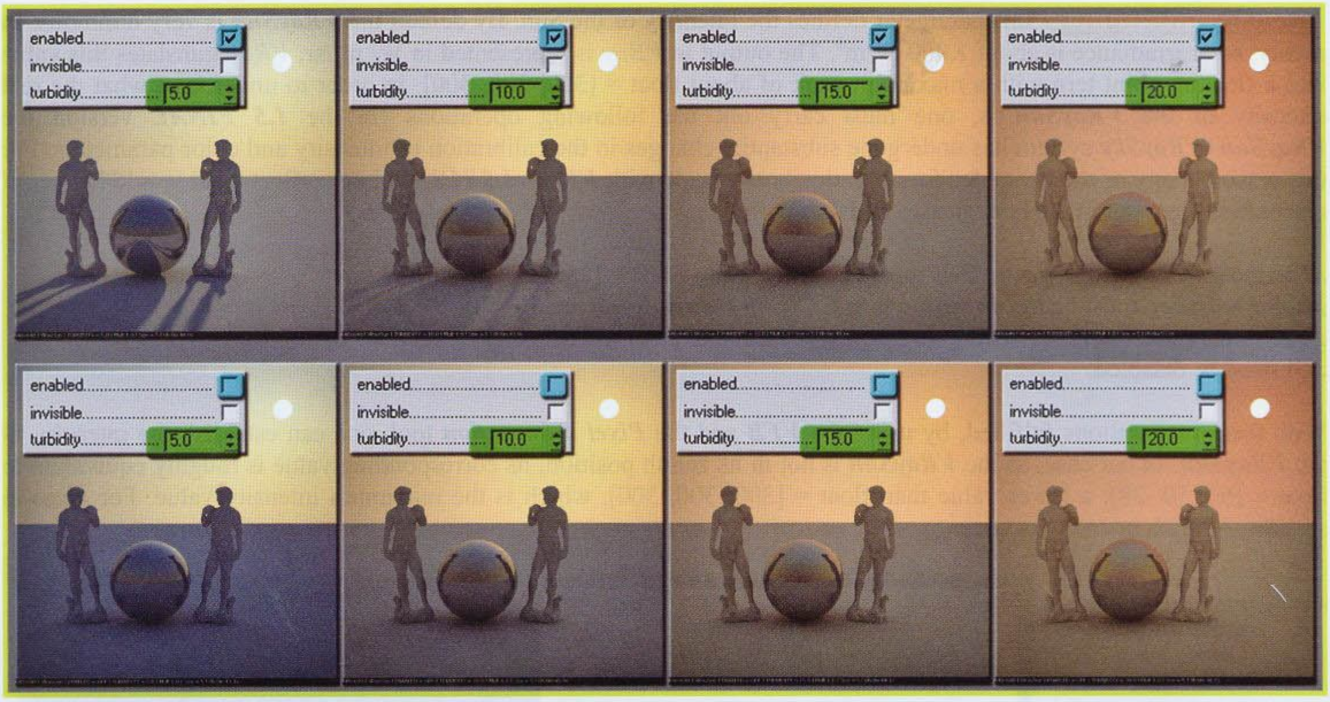
■ **Figure 6.262**
VRaySun invisibility. By activating the ***invisible*** option, the ***VRaySun*** disappears from reflective objects and from the background. Notice how for this test the radius of the sun has been increased to make it more visible.

turbidity - this parameter determines the amount of atmospheric dust in the atmosphere. This phenomenon influences both the ***VRaySun*** and the ***VRaySky***. Low values produce a clear sky with an intense Sun, typical of mountain areas. High values are useful to reproduce metropolitan environments with smog and dust. Therefore the sky appears yellowy-orange. The range of use goes from 2.0 to 20.0.



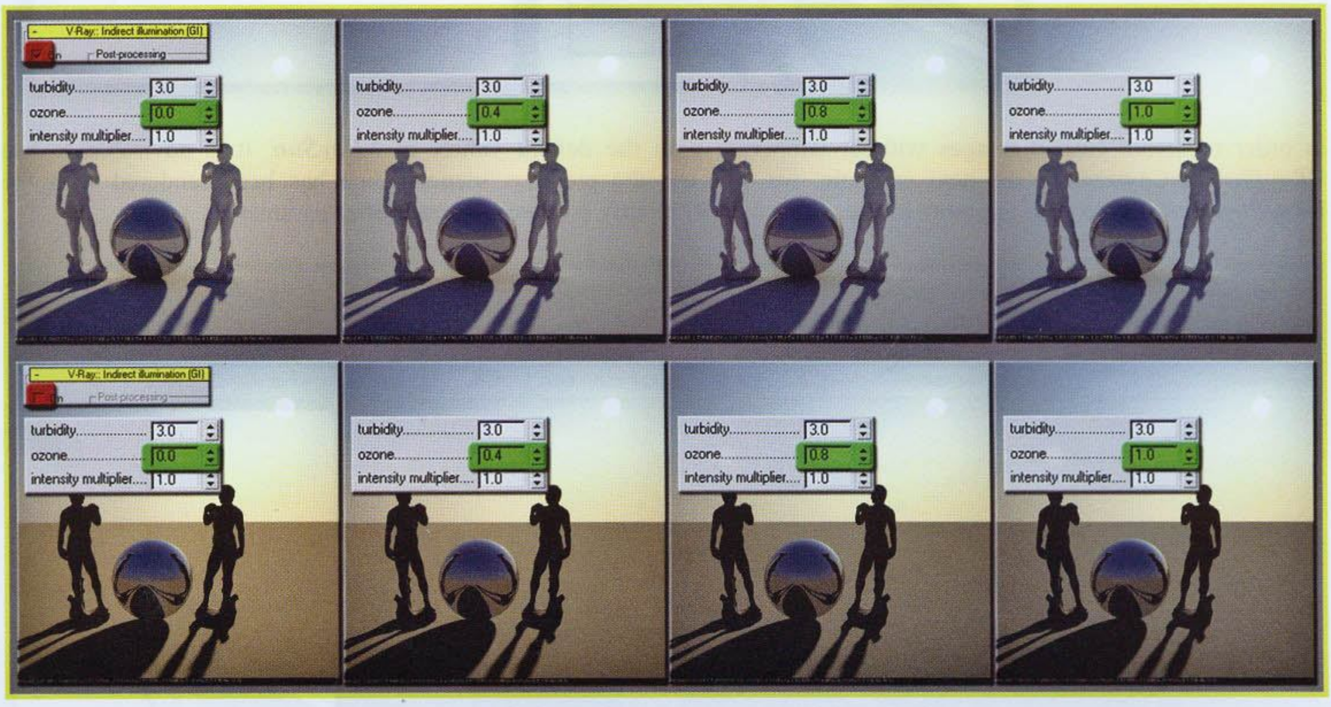
■ **Figure 6.263**
Variation of the ***turbidity*** of the ***VRayLight Sun*** light source. Notice how the color becomes more orange/pink as the value is increased.

Increasing the ***turbidity*** parameter, as well as changing the color of the sky, also reduces the influence of the ***VRaySun*** itself. This can be seen with high settings of ***turbidity***, beyond 10.0, the Sun-produced shadows tend to disappear. By comparing the tests with the ***enabled*** parameter on and off, one can tell that for high values of ***turbidity*** the Sun does not affect the scene at all, although it is active. It is as if the smog of the city were covering the Sun's rays which are unable to penetrate through the pollution cap caused by ***turbidity***.



■ Figure 6.264
Variation of the ***turbidity*** parameter with the ***VRaySun*** active and not.

ozone - this parameter influences the color of the light which is produced by the ***VRaySun*** only. Values near 0.0 produce a yellow color, instead color near 1.0 produce a blue/light blue color. This parameter goes from a minimum of 0.0 to a maximum of 1.0.



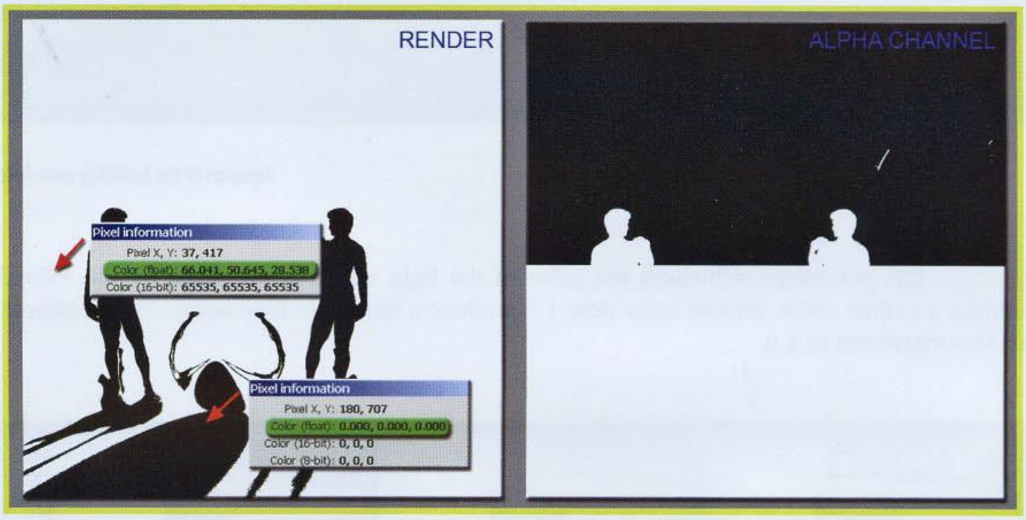
■ Figure 6.265
Variation of the ***ozone*** parameter with the ***VRaySun*** on or off.

From the tests one can tell that the influence of the ***ozone*** on illumination is quite low. Also, by changing only the light produced by the ***VRaySun***, Background does not change at all. Only the color of the light changes, and not that much either.We can see this from the plane representing the ground, which changes from yellow/brown to a rather neural grey. With the GI turned off, variations, although minimal, are much more visible.

intensity multiplier - this parameter defines the intensity of the Sun. By default the ***VRaySun*** is very luminous. In reality, solar irradiance is about 1,000 W/m^2. The output in ***VRay*** is represented in W/ m^2/sr. ***VRay*** translates this value into a floating-point tern with a maximum peak of about float = [300, 300, 300]. In order to understand what the real intensity of the ***VRaySun*** is, one must carry out the following operations (in the ***1.5 FINAL*** version the ***VRaySun+VRaySky*** system has undergone substantial changes in the calibration of intensity and color parameters. For this reason, many of the examples found in this book, made with ***VRay v1.5RC3***, will not reflect what would happen if one carries them out in the last releases of ***VRay***).

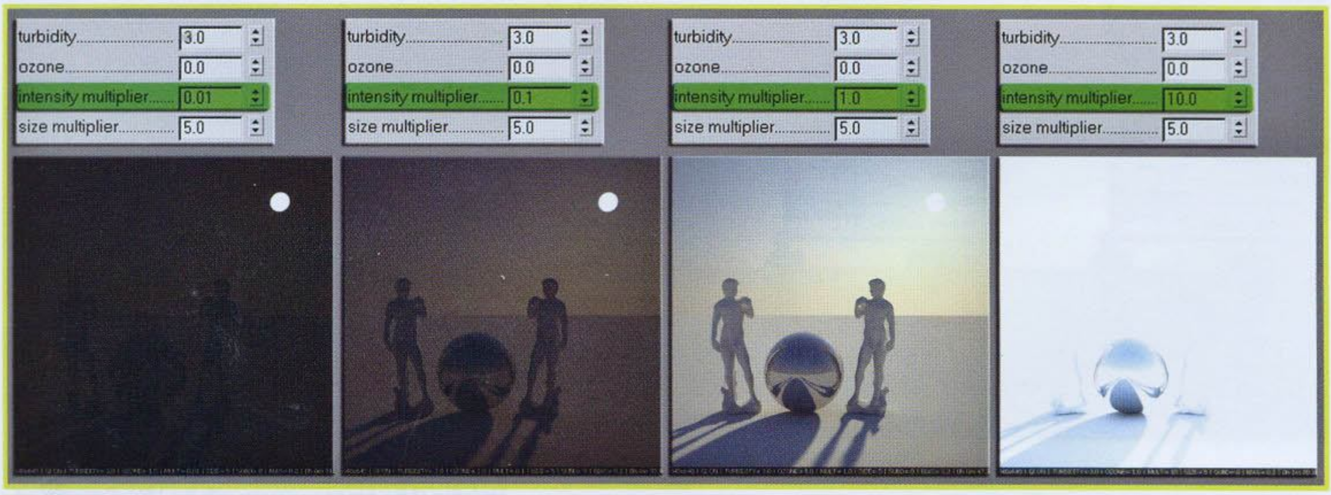
. The material applied to geometries must be a pure white RGB = [255, 255, 255].
. ***Color mapping*** must be in ***Linear*** mode and ***Clamp output*** deactivated.
. The camera must have ***vignetting*** and exposure disabled, so as to avoid any corrections of exposure.
. GI must be disabled.

With these preparations fulfilled, by using the ***VFB*** and the ***Pixel information*** tool, one can establish the intensity of the ***VRaySun***. In our case, as the ***VRaySun*** is not in its zenith position, its corresponding value is roughly equivalent to float = [66, 50, 28], a lower value than float = [300, 300, 300], which is the maximum intensity value. For areas in shadow, instead, intensity is none.



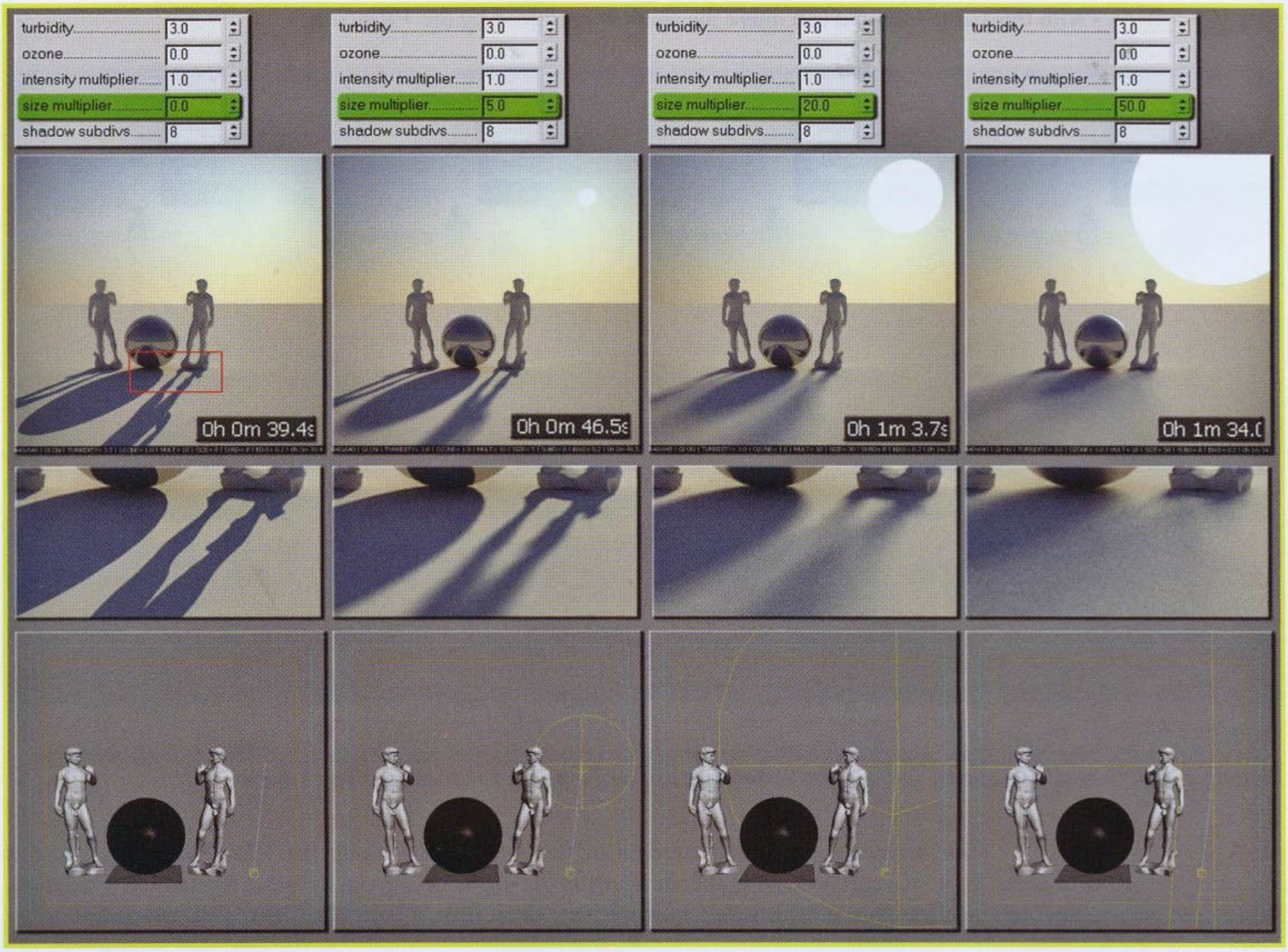
■ **Figure 6.266**
In this example, the intensity of the sun and the luminous flux produced by it is about float=[55,50,28]. Such an overexposure must be controlled and countered by a correct exposure, obtained with a ***VRayPhysicalCamera***.

In order to create correct images without tampering with the default values of ***VRaySun***, it is advisable to use ***VRayPhysicalCamera***. In the next example, we will use the previous scene, which has been rendered with the ***VRayPhysicalCamera*** with exposure active. After we will modify the ***intensity multiplier*** parameter.



■ **Figure 6.267**
Change in ***VRaySun*** intensity. The value to be used for the best results is without a doubt ***intensity multiplier*** = 1.0.

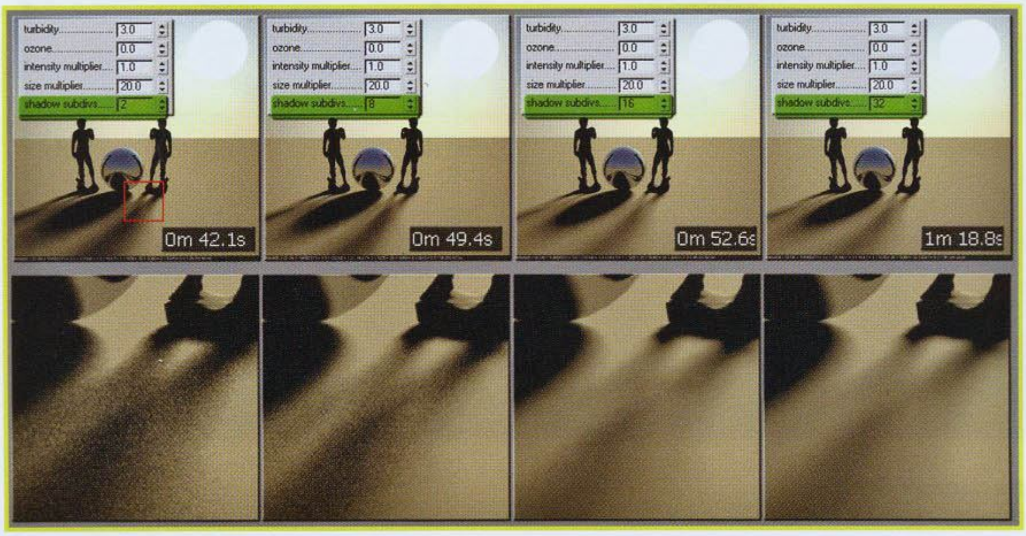
size multiplier - this parameter changes the size of the Sun. In the scene it determines the size of the solar disc both in the camera and in reflections, and also the size of ***Area shadows***. High values correspond to a greater radius and blurrier shadows. Values nearing 0.0 make the Sun much smaller, with defined shadows. Like all ***VRay's*** light sources, the quality of blurring is controlled by ***subdivisions***.



■ **Figure 6.268**
Variation of the size of the Sun, with consequent action on shadow smoothness.

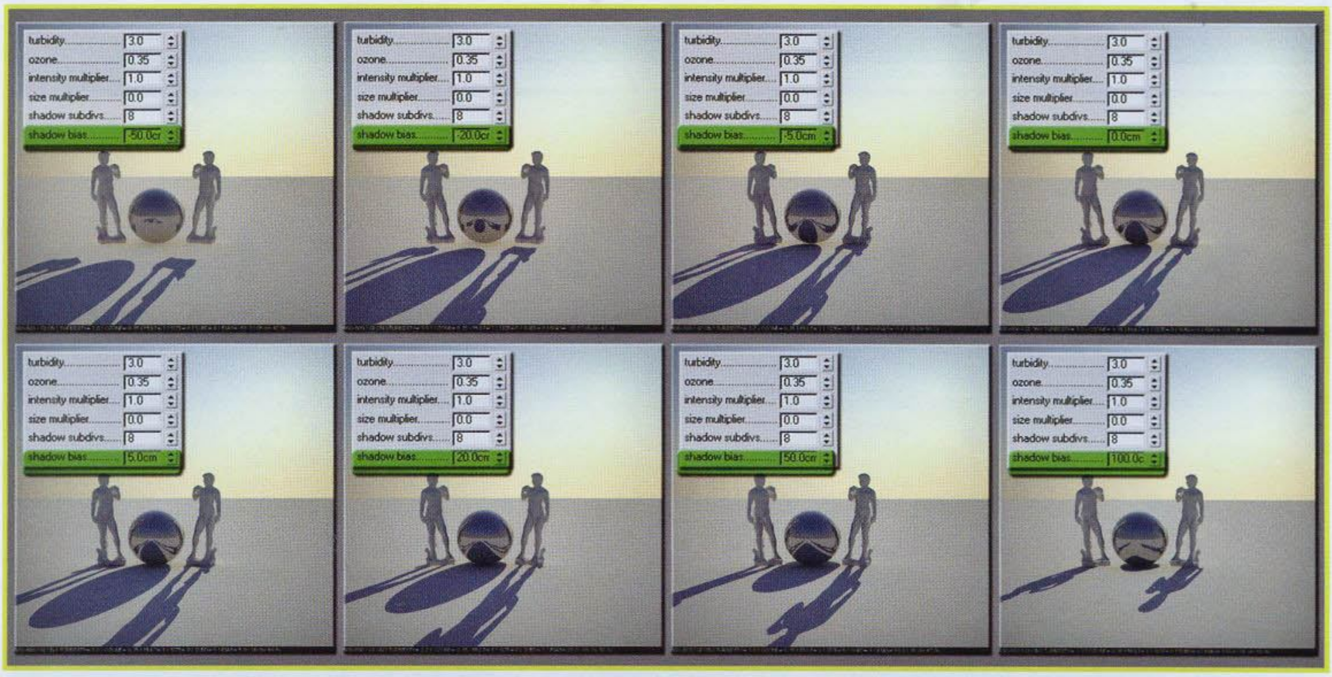
With a setting of 0.0 one obtains a very defined shadow, typical of classic raytrace shadows, whereas higher values produce much blurrier shadows. The increase in rendering time is inevitable, due to the very fact of ***Area*** type shadows being present.
Another characteristic which is interesting is the size of the ***VRaySun*** icon. Its size changes with the ***size multiplier*** parameter. The greater the ***size multiplier***, the greater the radius of the icon. This gives the user a first approximation of what the size of the Sun will be in the final render.

shadows subdivs - this parameter allows one to control the quality of shadows produced by the ***VRaySun***. High settings create better quality shadows with less noise, but rendering time increases. Low settings decrease rendering time, but shadows have more noise.



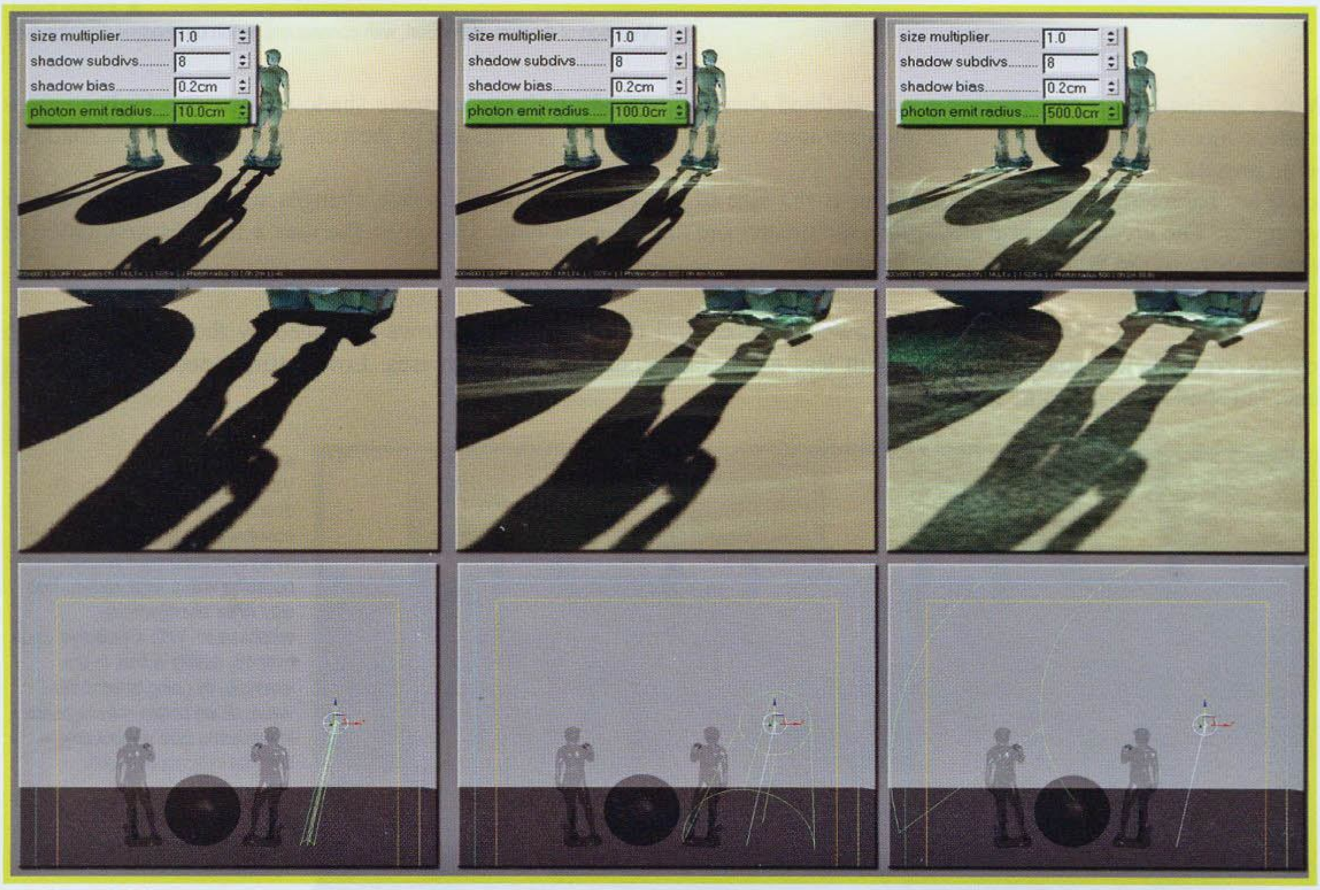
■ **Figure 6.269**
Variation in shadow quality. By deactivating the GI, one can observe the noise produced by ***VRaySun*** better, set for the occasion with a large radius. This way ***Area shadows*** are emphasized. With a ***Subdivs*** value over 16, quality is fine. In the example, by going beyond this value, all we obtain is an increase in rendering time without any benefits in terms of quality.

shadows bias - with this parameter one can move the shadow produced by the ***VRaySun***.



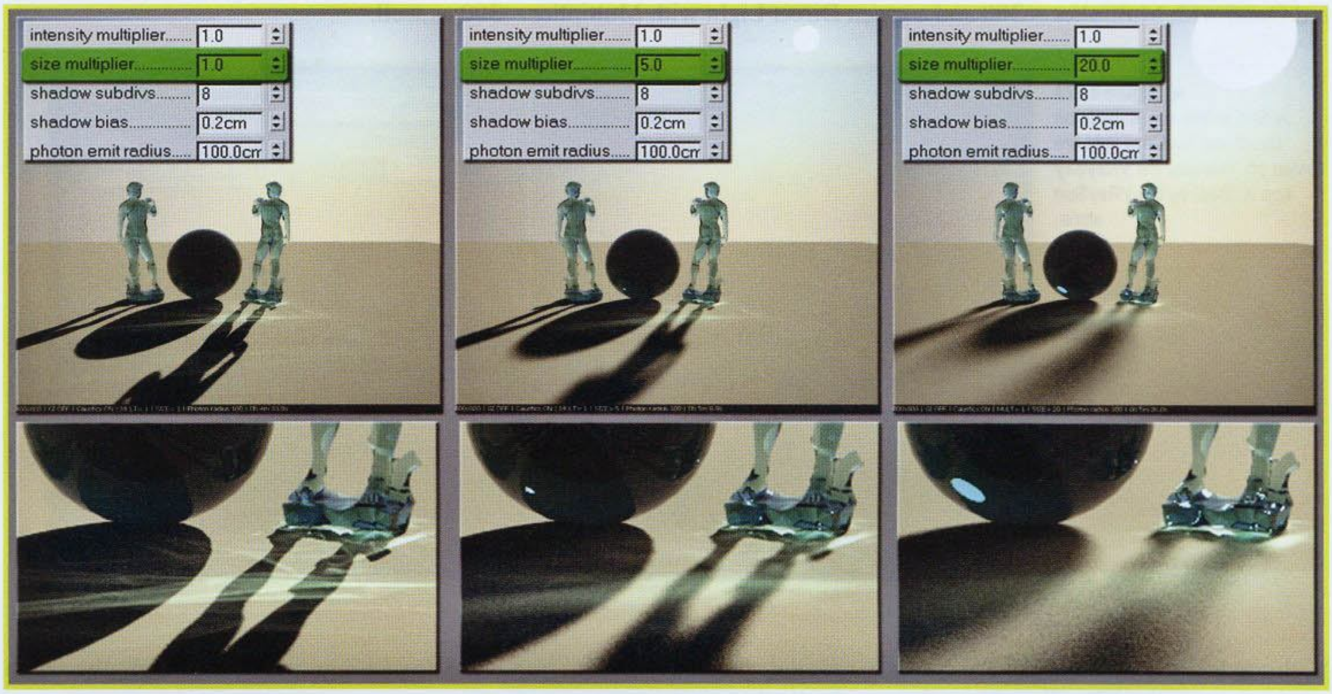
■ **Figure 6.270**
Change in position and size of the shadow generated by the ***VRaySun***. With negative values the shadows move further away from the models until they reach the point whereby the mesh and the shadow no longer coincide, instead positive values tend to squash the shadow, making it shorter.

photon emit radius - this parameter controls the radius of the area within which the ***VRaySun*** emits photons dedicated to caustic effects. This area can be seen in the viewport in the shape of a green cylinder around the ***VRaySun***. Remember that in order to visualize caustic effects, one must activate them from within the ***VRay: Caustics*** rollout.



■ **Figure 6.271**
Effects due to the variation of the radius of emission of photons.

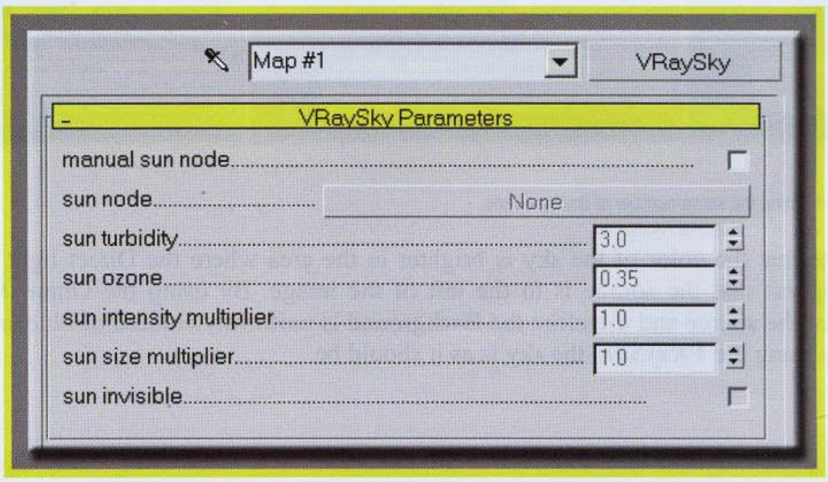
With an excessively small radius, visible also from the viewport, the photon cone does not reach any of the elements which could generate caustic effects. With a value of 100, photons are able to collide with the right-hand statue, generating good-quality caustic effects. If one keeps the number of photons the same, managed via the ***VRay light properties*** panel, and increases their radius of influence, the ***VRaySun*** is able to reach the other geometries as well. In this case the caustic effects produced are of a lower quality compared to the previous ones. This is because ***VRay*** has to cover a wider surface with the same number of photons. Therefore quality is necessarily reduced: caustic effects are less precise.
Also the size of the ***VRaySun***, which can be changed with the ***size multiplier*** parameter, affects the caustic effects directly. The greater the radius of the solar disc, the less precise and the blurrier caustic effects will appear.



■ Figure 6.272
Blurry caustic effects caused by increasing the size of the ***VRaySun***.

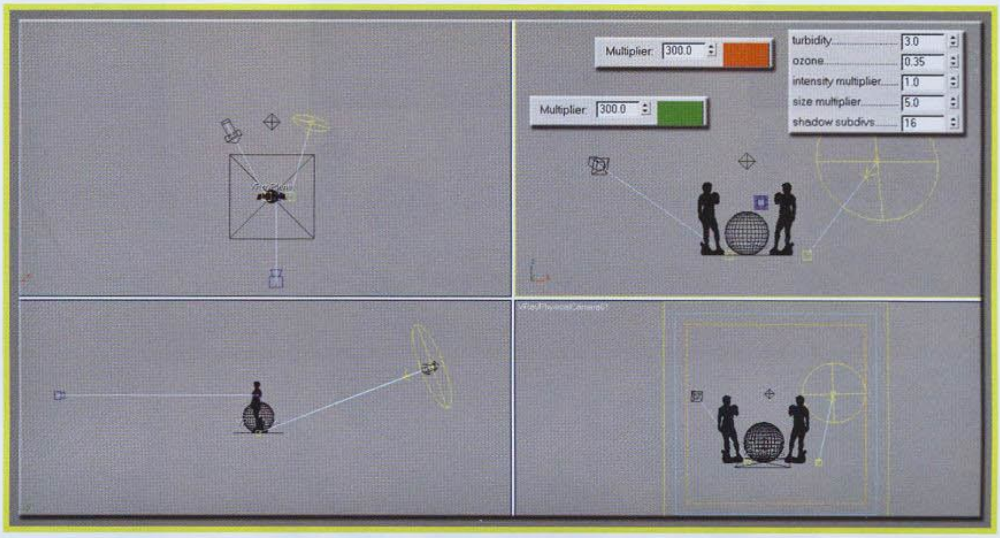
PARAMETERS VRaySky

The ***VRaySky*** map is used for simulating the sky. The ***VRaySky*** texture is generated and inserted automatically in the 3ds Max Environment when one creates a ***VRaySun***, on which it depends. It depends on it in the sense that by changing the position and parameters of the ***VRaySun***, the ***VRaySky*** map is automatically changed: a handy commodity! If for instance the Sun is low on the horizon, the sky will accordingly take on a reddish color. If the Sun is high up, the sky will be bright blue. ***VRay*** takes care of all this, as it is the program that takes care of the synergy between ***VRaySun*** and ***VRaySky***. The user need only move the ***VRaySun*** where he prefers, nothing more.



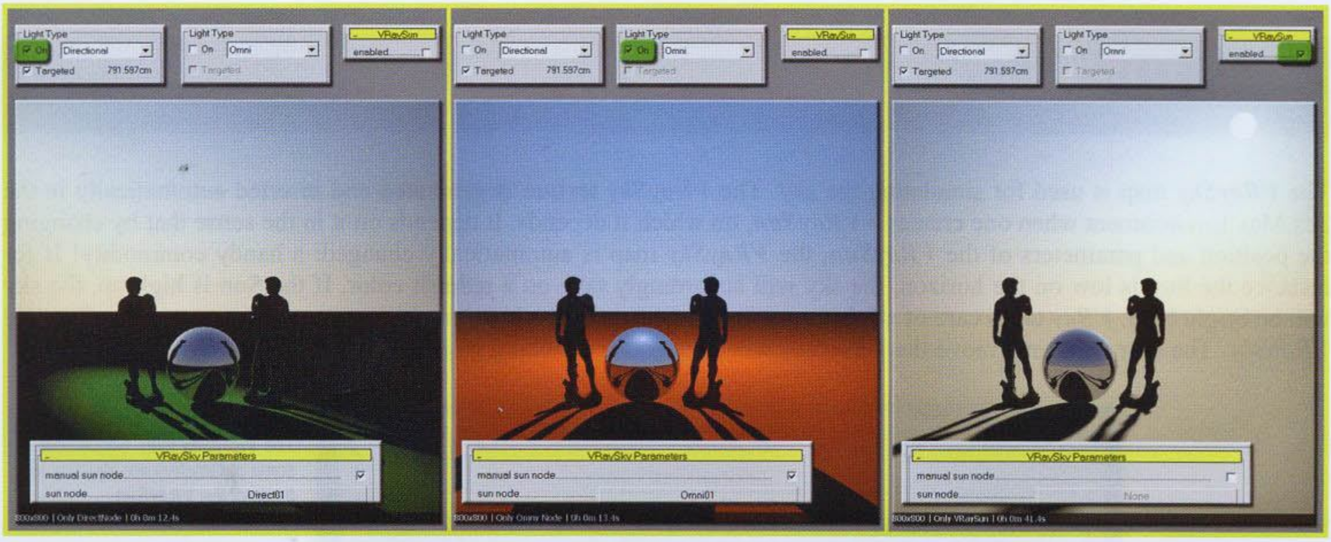
■ Figure 6.273
The ***VRaySky*** map. Notice how some parameters are identical to those of the ***VRaySun***.

manual sun node - this specifies the way the ***VRaySky*** map determines its parameters, color and intensity. If it is disabled, as it is by default, sun ***turbidity***, ***ozone***, ***multiplier*** and so on are defined by the ***VRaySun*** automatically, according to its position and parameters. If it is activated, one can choose a new light source. This way the texture and its direction are managed by the new light source.
If one observes the sky on a nice day, it is brighter, clearer and whiter nearer the Sun, and darker in the areas which are further away. By creating a new light and choosing it as the new "Sun", the ***VRaySky*** parameters are no longer controlled by the ***VRaySun***, but by the new Direct Light. The choice of a Direct Light is not casual. It is in fact preferable to choose a directional source as opposed to an omnidirectional light, because the former has a vector which allows *VRay* to calculate the appearance of the ***VRaySky***.
In the following scene, three sources have been placed. From right to left, a ***VRaySun*** with ***intensity multiplier*** = 1, a red Omni with Multiplier = 300 and a green Direct Light with Multiplier = 300 as well.



■ **Figure 6.274**
At first, in this scene, both the Direct and the Omni light are turned off. This way the ***VRaySky*** map is "lead" by the ***VRaySun*** alone.

With the GI still deactivated, the scene is calculated first of all with only the Direct Light active, then only the Omni and lastly only the ***VRaySun***. Now observe the Backgrounds.



■ **Figure 6.275**
Three different luminous sources determine the same number of Backgrounds.

In the first case one can see that the color of the sky is brighter in the area where the Direct light source is located. There are blends which suggest that the source is to the left of the image. By using the Omni, *VRay* is unable to establish the directionality of the source and therefore the Background is uniformly colored of blue, as one can also see from the chrome sphere. By using the ***VRaySun***, the sky is as it should be.

It is also possible to have two or more sources in the same scene, for instance a Direct Light and the ***VRaySun***. In the next example we will use the Direct Light as the sun node for ***VRaySky*** but the ***VRaySun*** will be active as well.

■ **Figure 6.276**
Use of two light sources, as well as the ***VRaySky*** map.

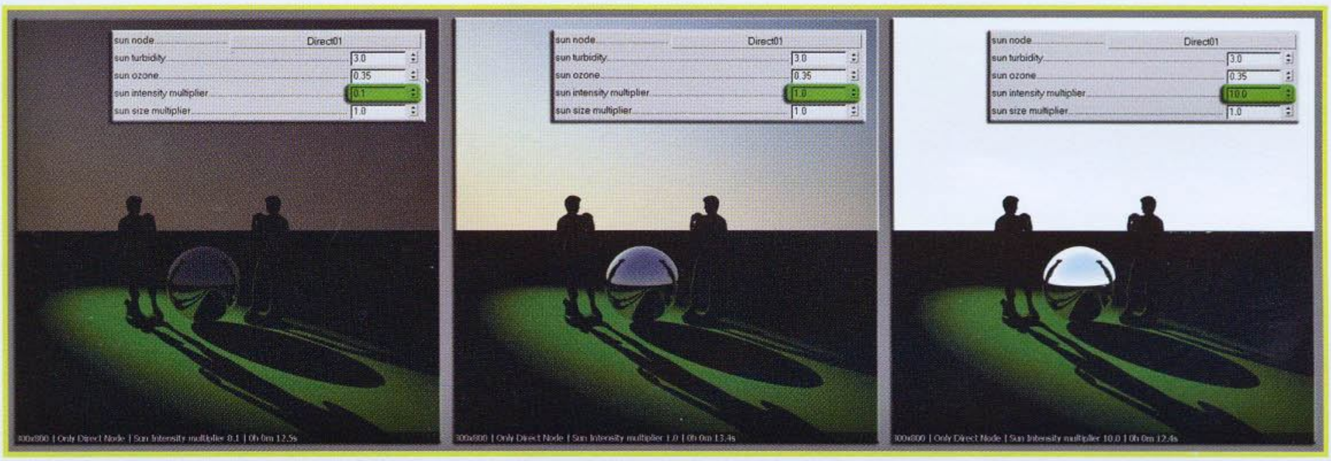
Using ***VRaySky*** together with a Direct Light which works as a node for the ***VRayMap*** does not represent a problem. The only defect is due to the creation of a double shadow. Background is managed by the position of the Direct Light in any case.

sun node - with this button, one can specify the new sun, as an alternative to the ***VRaySun***.

sun turbidity - after having activated the ***manual sun node*** parameter, all the parameters of the ***VRaySky*** become modifiable, amongst which also the ***sun turbidity*** parameter. As on is using an alternative source and not the ***VRaySun***, like for example a Direct Light, and because the latter does not have the same parameters as the ***VRaySun***, ***VRay*** provides the ***VRaySun*** values inside the ***VRaySky map***, which are used for changing the ***sun node***, in this case, of the Direct Light. ***Turbidity*** is a parameter which we have already covered for ***VRaySun*** and so one can refer to the previous explanation.

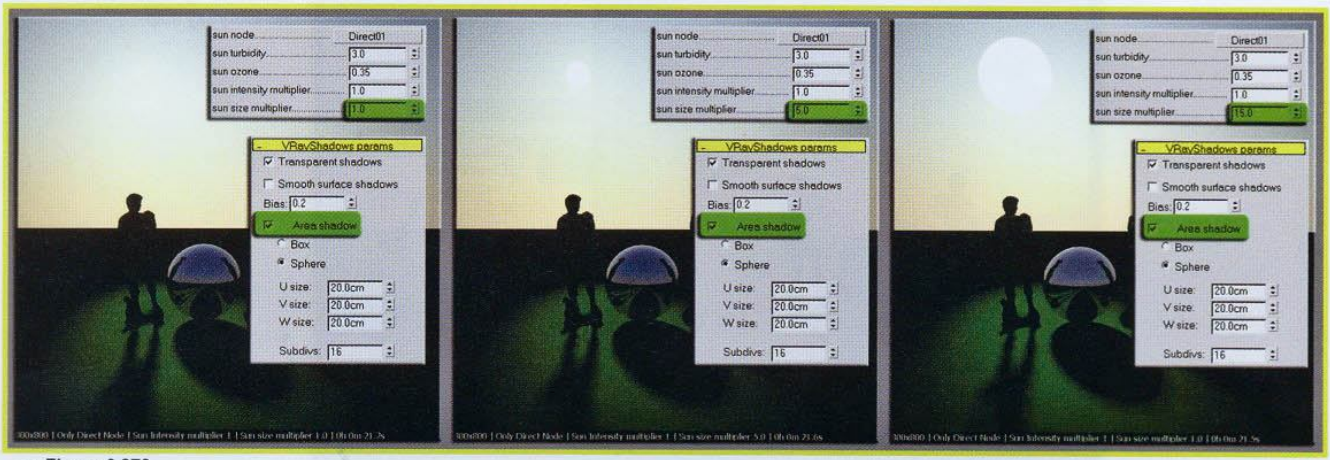
sun ozone - like the previous parameter, ***sun ozone*** controls the amount of ozone in the atmosphere, using a light source which is no longer ***VRaySun***, but a different light.

sun intensity multiplier - this controls the intensity of the new light source. One must be careful, as this parameter does not change the multiplier of the Direct Light, but only the luminosity of the sky. In order to change the Sun's ***multiplier***, represented by the Direct Light, one must change the 3ds Max light parameters directly.



■ **Figure 6.277**
None of the parameters in the ***VRaySky*** influence the Direct Light in any way.

sun size multiplier - like for ***VRaySun***, the ***sun size multiplier*** changes the size of the solar disc. In this case, shadows and their smoothness are not influenced. In order to change ***Area shadows***, one must act upon the Direct Light directly, via the parameters located in the ***VRayShadows params*** rollout.



■ **Figure 6.278**
Dimensional variation of the solar disc due to the change in the ***sun size multiplier*** parameter.

sun invisibile - as is true for ***VRaySun***, the ***sun invisible*** regulates visibility on reflected objects or from the direct viewing.

CHAPTER 7: OBJECT and ENVIRONMENT

Until now numerous aspects of VRay have been considered, from common parameters to many other rendering engines. Shaders, lights, and settings for the GI are elements which can be found in any Renderer. VRay has many other features. Some of these are contained both in the Object category and in the section devoted to the Environment effects.

■ Figure 7.1
The four objects available in VRay and the two atmospheric effects present in the Environment panel.

Throughout the book, each example shall be explained and illustrated. For now it is enough to know that VRay's objects can be defined as "generators of geometries", and can be selected and rendered, while the atmospheric effects are FX, special effects enabling interesting results.

VRay adds four new objects within 3ds Max. They are located in the Geometry category, a subsection of VRay, and allow one to create very particular geometries.

■ Figure 7.2
VRay Objects are located in the Geometry panel of 3dsMax.

CHAPTER 7: OBJECT and ENVIRONMENT

VRAY: OBJECT and ENVIRONMENT

Until now numerous aspects of *VRay* have been considered, from common parameters to many other rendering engines. Shaders, lights, and settings for the GI are elements which can be found in any Renderer. *VRay* has many other features. Some of these are contained both in the *Object* category and in the section devoted to the *Environment* effects.

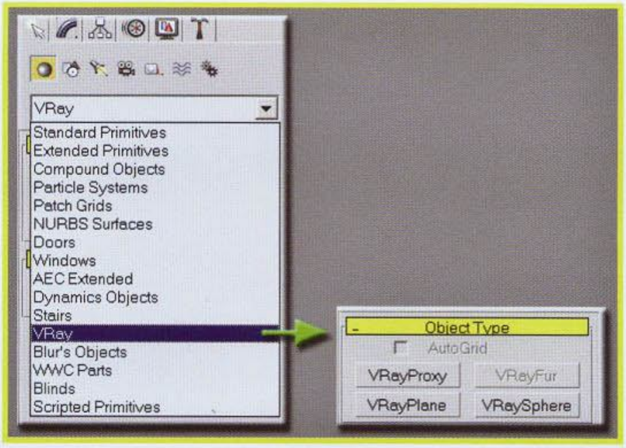
■ **Figure 7.1**
The four objects available in VRay and the two atmospheric effects present in the ***Environment*** panel.

Throughout the book each example shall be explained and illustrated. For now it is enough to know that *VRay*'s objects can be defined as "generators of geometries", and can be selected and rendered, while the atmospheric effects are FX, special effects enabling interesting results.

VRay Objects

VRay adds four new objects within 3ds Max. They are located in the Geometry category, a subsection of *VRay*, and allow one to create very particular geometries.

■ **Figure 7.2**
VRay's ***Objects*** are located in the Command panel of 3ds Max.



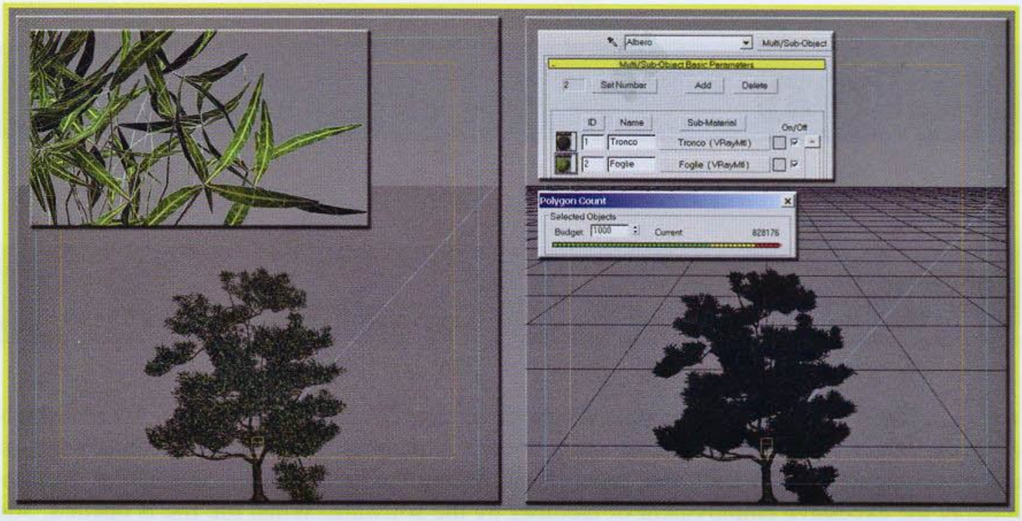
VRayProxy

INTRODUCTION

Imagine a forest of 1,000 trees completely modelled in 3D. Surely it is a heavy scene, which might contain millions of polygons. Many PCs are not able to support such a quantity of geometries. Some crashes of the system may occur during the opening of the files and, most of all, during the computing of the render. In these cases the reason is most probably the excessive employment of memory by the software. When the file is opened, 3ds Max searches all the data of the scene in the memory. This affects the stability of the program and normally leads to an untimely saturation of hardware's resources before the rendering computation.

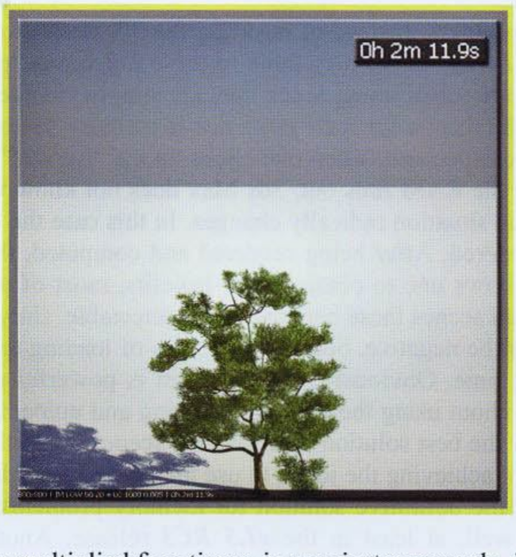
VRay provides a very cleaver solution to this problem. Instead of inserting the 3D models of the trees inside the .max file, it positions some substitutive geometries, ***Proxies***. One may think to these objects as "placeholders" which, when necessary, are replaced by the geometries they are substituting. Since they are simply "markers", the .max file does not weigh much. The appearance of ***Proxies*** and the reason for their existence is memory saving. Often 3ds Max crashes because of a saturation of the RAM. This happens because, when one clicks on the "Render" button, 3ds Max sends all the geometries directly to the memory. In case the RAM runs out, 3ds Max does not know where to put the data sent, and it seizes up. Instead, by using the ***Proxy***, the situation radically changes. In this case the external files, the ***Proxies***, are loaded on memory only when they are rendered. After being rendered and computed, the memory is deleted thus leaving the space for new information. This allows one to obtain many benefits, most of all for heavy files, such as complicated geometries to be rendered. For small scenes these benefits are undetectable, since the RAM is itself enough for the render. The use of the ***Proxy*** might even be negative, because a process of loading/unloading of memory might occur, leading to an increase of the rendering time. Obviously, if the system is powerful enough, it is preferable to verify if 3ds Max is able to render the scene without using the ***Proxies***. Loading and unloading the RAM continuously generates a process which requires time. Surely the best solution is to load the scene all together in the memory. In case the polygons are too numerous, the only way for achieving the job is to use the ***Proxy***. In order with what has been said up to now one might think the ***VRayProxy*** is the definitive solution for memory problems due to a high number of polygons. Unfortunately its has its faults as well, at least in the ***v1.5 RC3*** release. Another problem might arise, concerning the possible similarity between the ***VRayProxy*** and the ***XRef***. In 3ds Max an ***XRef*** element is an external reference file. A scene might contain many ***XRef*** referring to other objects or other scenes. ***XRef*** objects allow several animators or modellers to operate simultaneously on a scene without interfering with each other's job. Unfortunately, at the moment of the rendering, the ***XRef*** are always completely loaded on memory. Therefore, the reason of the introduction of the ***VRayProxy*** is clear.

Before analyzing ***Proxy*** parameters, a brief tutorial shall be carried out in order to better understand how they work, how they are fulfilled and how the ***VRayProxies*** are applied to a real scene.
A tree completely modelled with polygons, leaves and branches included, is used as reference mesh. The total weight of this geometry reaches 830,000 polygons, nearly 1 million! It cannot certainly be defined a light model. The whole of this leans on a simple plane, illuminated by means of a ***VRaySun+VRaySky*** system, and observed by a ***VRayPhysicalCamera***. The 3D model is completely textured through a Multi/sub-object composed by 2 ID.



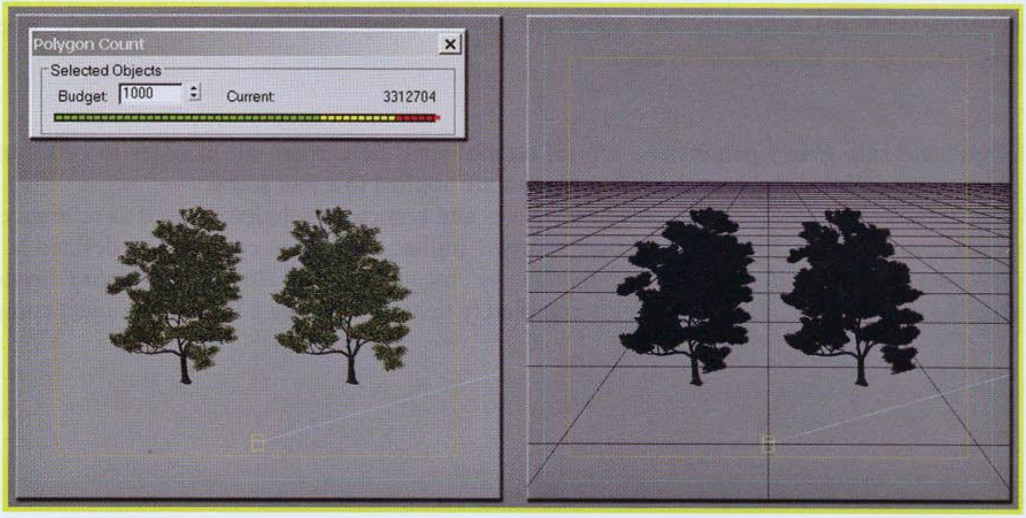
■ **Figure 7.3**
The scene to be used in the following tests on ***VRayProxy***. In this case the geometry is physically present on the scene.

The rendering is carried out in very few seconds without particular problems.



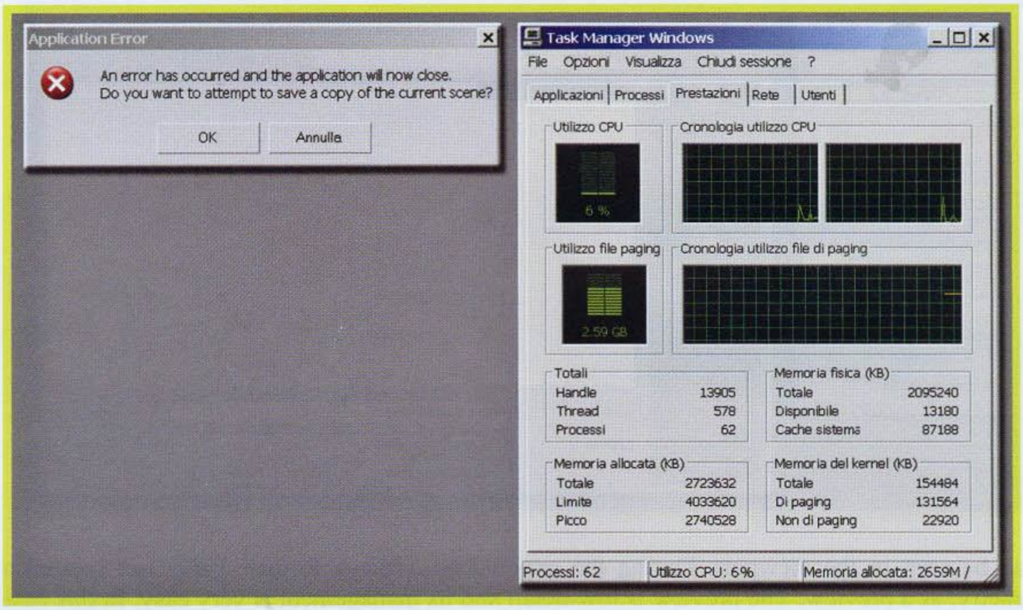
■ **Figure 7.4**
Render of the scene by using the 800,000 polygons mesh.

Now the tree is going to be multiplied four times, in non-instance mode, in total reaching 3,312,704 million polygons.



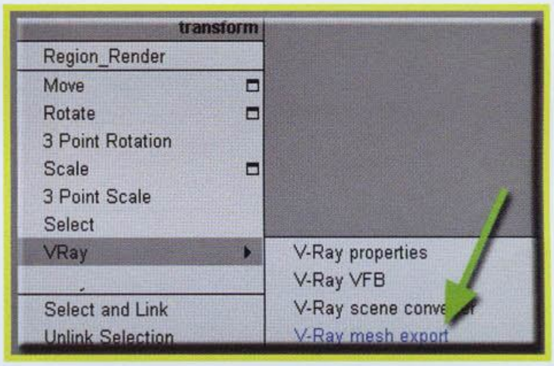
■ **Figure 7.5**
Increase in the scene's complexity generated by having multiplied the geometry four times. The trees seem to be two, but are in truth four overlapped.

Unfortunately, with the system used for rendering, having 2 GB RAM, it has not been possible to achieve the render due to the excessive demand of memory, as shown by the untimely closedown of the program and by the indication in the Task manager of the memory used.



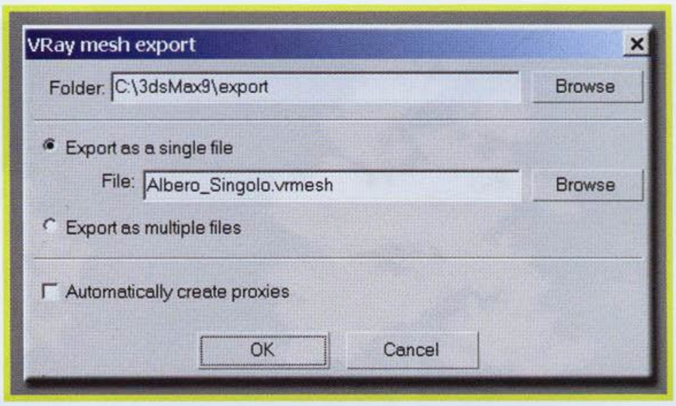
■ **Figure 7.6**
Untimely closedown of the software due to an excessive use of RAM.

At this point one has to use the ***VRayProxy***. The procedure for creating one of them is very simple. It is enough to select the geometry one wants to use as ***Proxy***, in this case the single tree, and click on the option ***VRay mesh export*** present in the Quad Menu.



■ **Figure 7.7**
In 3ds Max Quad Menu several commands are to be found, among which the one for exporting the mesh to be used as a ***VRayProxy***. To be noticed how the Quad Menu has been slightly modified by the author.

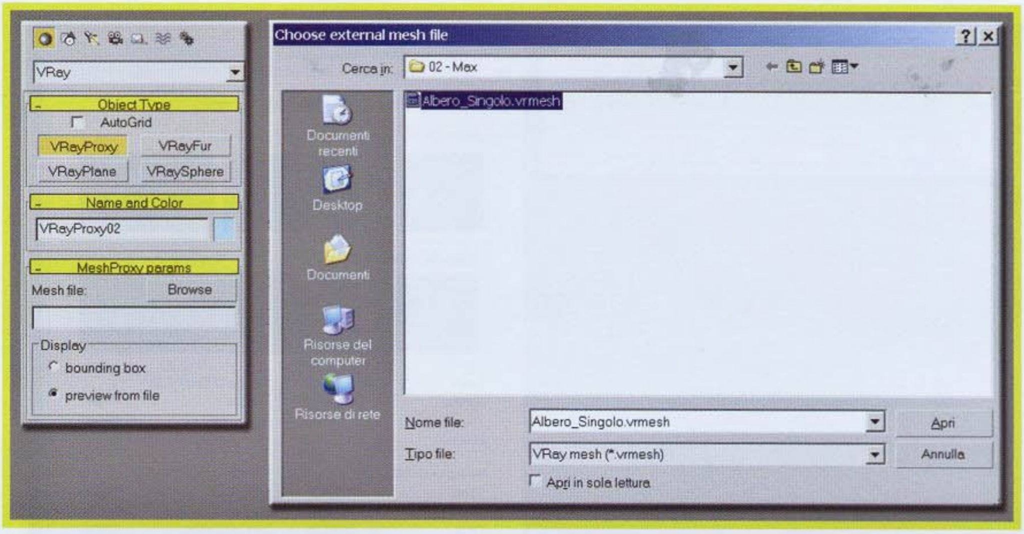
Afterwards a table appears, allowing one to export the geometry in the 3D proprietary format of ***VRay***, the ***.vrmesh***. This file contains all the geometric information about the mesh which has just been exported, such as vertexes, edges, texture channel, material ID, smooth groups, etc… that is, everything necessary for rendering.



■ **Figure 7.8**
The ***VRay mesh export*** table is a window allowing one to create the ***.vrmesh*** owner file.

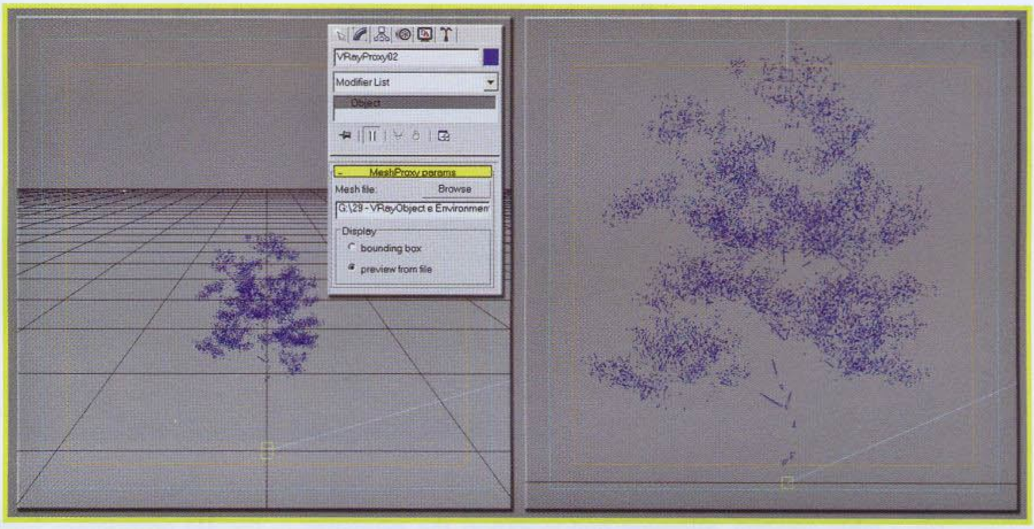
Later, in the book, all the values of the ***VRay mesh export*** tools shall be analyzed. For now it is enough to know that, through it, a 3D file ***.vrmesh*** is created in a certain position in the hard disk which is to be used as a ***VRayProxy*** object. Now, by clicking on the ***VRayProxy*** button before, and then in the viewport, a new window appears with which it is possible to insert any ***.vrmesh*** file. In this example it is the tree which has just been exported.

■ **Figure 7.9**
Creation of a ***VRayProxy*** object.



Finally the ***Proxy*** object can be seen in the viewport. During the export into ***.vrmesh*** format, *VRay* has created a preview of the exported geometry, as a preview. The preview is low-detail and its management is very easy. In this case it is a simplification of the tree.

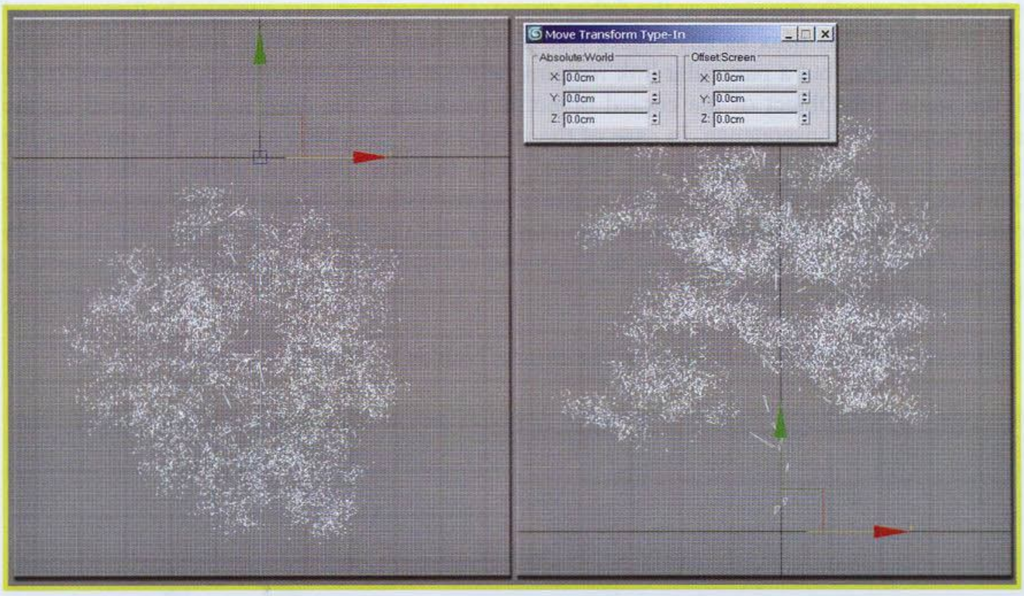
■ **Figure 7.10**
Perspective and frontal views of a ***VRayProxy*** object.



It is worth citing the position of the Pivot of a ***VRayProxy*** object. Observe the original scene, and the position of the Pivot compared to the World coordinates [0, 0, 0].

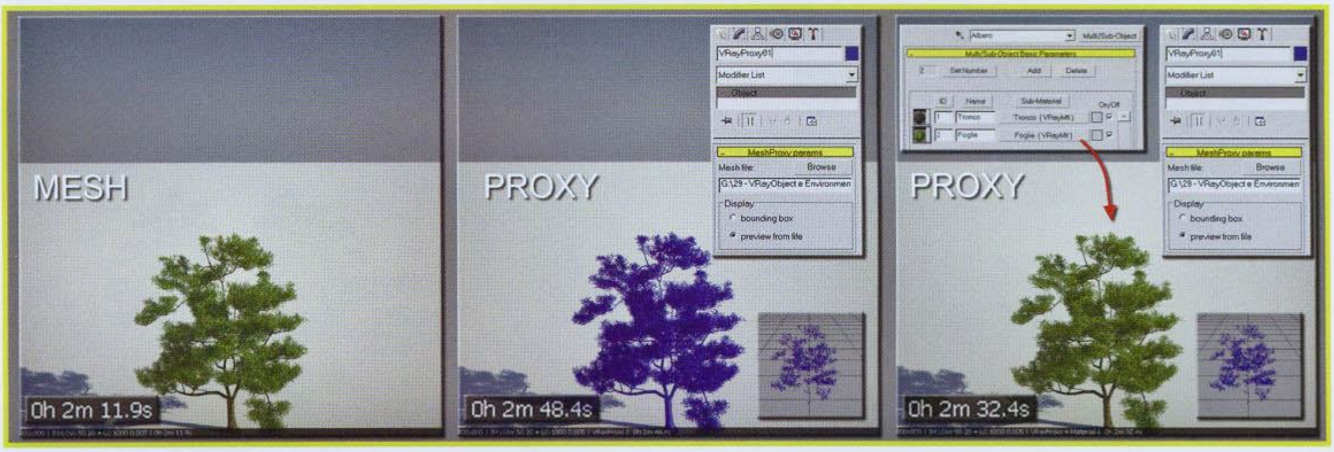
■ **Figure 7.11**
The tree's Pivot is located exactly at the foot of the tree-trunk, which is a little further from the origin of the global coordinates system.

By exporting the mesh, the Pivot of the ***VRayProxy*** is going to be placed at the coordinates [0, 0, 0] and no longer at the foot of the trunk. This way it is very easy to insert it in the same position of the original mesh. In fact, it is enough to align the ***Proxy***'s Pivot to the coordinates [0, 0, 0], and the object of *VRay* is placed in the same location.



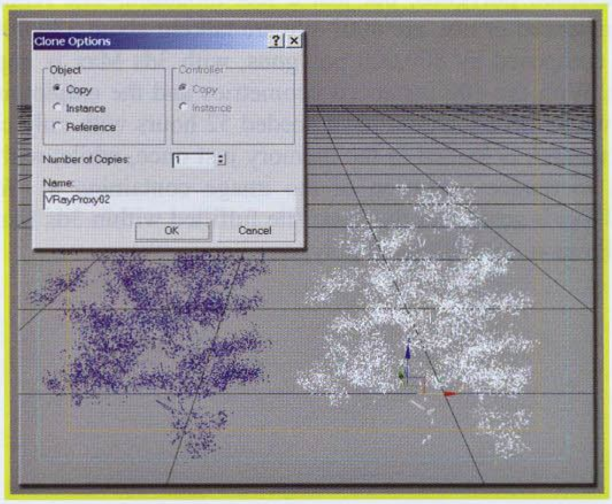
■ **Figure 7.12**
Since the ***VRayProxy*** has been created when the original object was not yet aligned to the centre of the global coordinates, the ***VRayProxy*** and its Pivot are not aligned. This allows one to insert it correctly in the same position of the original one, by using as reference the centre of the global coordinates.

Obviously it is necessary to align the object with the centre of the global coordinates prior to the exportation of the ***.vrmesh*** to create a ***Proxy*** object with the pivot aligned to the original one.
Now almost everything is ready for rendering. The ***Proxy*** is correctly inserted and visible in preview. But the first render is completely blue, as is the Wire color of the ***Proxy***. It is clear ***Proxies*** do not contain any information concerning materials. Therefore, it is necessary to assign one to them. In this case the Multi/sub-material of the original tree.



■ **Figure 7.13**
A single ***VRayProxy*** correctly rendered.

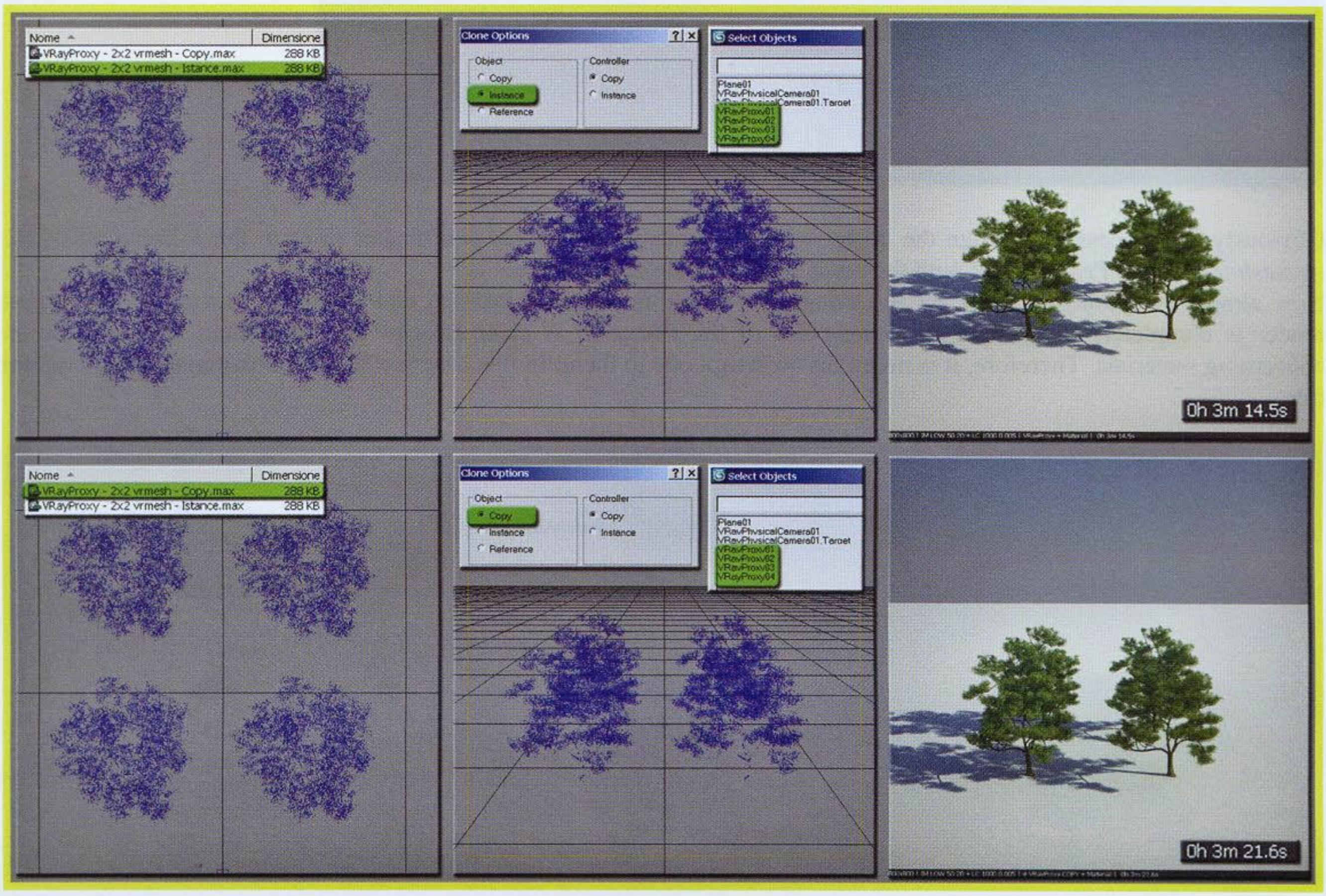
Before proceeding a brief explanation concerning Copies and Instances is needed. When one copies any object in 3ds Max, it is possible to choose between Copy, Instance or Reference.



■ **Figure 7.14**
When copying an object, 3ds Max offers three solutions: Copy, Instance and Reference.

By selecting Copy, the copied object is exactly the same as the father object. Once copied, it is completely disconnected from the original one and it is treated by 3ds Max as any object would be. Instead, by selecting Instance, the copied object is not just the same as the parent one, but a particular link remains between them. For example, if the vertexes of the father object are moved, the same Sub/Object of the copied object is modified. This does not only speed up the modifying operations, but has an important consequence. In case of the simple copy, 3ds Max stores in the memory the two objects as completely foreign meshes even though similar. Instead, with Instance, 3ds Max stores in the memory just an object, the parent, from which it obtains all the information needed to compute the children at the moment of the rendering. This produces a relevant effect on the management of the memory and the RAM. Always for this reason, in the previous example it has been chosen to copy the four trees as Copy and not as Instance, in order to use, as early as possible, all the memory of the PC and stress 3ds Max.

Always referring to the previous scene, observe the behaviour of the objects both as Instance and Copy.



■ **Figure 7.15**
Visualization in The viewport and rendering of ***VRayProxy*** cloned by the options Instance and Copy.

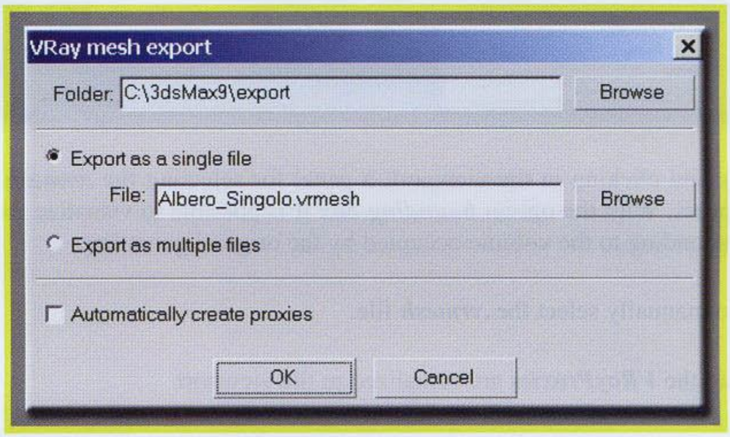
The size in KB of the two .max files are the same, and are low as well, since inside the scene there are not four geometries composed by 800,000 polygons, but just a few thousand. The rendering time is nearly the same. What happens when the copies, instanced or not, are 400? We are talking about 330,000,000 million polygons! Expecting to render 400 trees, each one composed by 800,000 polygons, with 3ds Max is impossible, but it is not a problem with *VRay* and its ***VRayProxies***. With the 400 instanced geometries and the employment of the ***IM+LC*** system, rendering time is roughly 9 minutes, while the ***LC*** would have needed 52 hours with only copied images! This happens because, by using Instance, the file ***.vrmesh*** is loaded in the memory just once, while, with Copy, *VRay* recalls the external file ***.vrmesh*** each time it tries to render a portion of the image containing a ***VRayProxy***. All this has been said to demonstrate how not only *VRay*'s setup, but the same file fulfilled within 3ds Max as well, has to be set in such a way to allow *VRay* to render millions if not billions of polygons.

PARAMETERS

Two simple operations must be carried out with the *Proxy*: the creation and export of the geometry to be "proxied" in the owner format *.vrmesh*, and its introduction in the scene through the *VRayProxy* object. *VRay* uses two windows for these operations. A preview of them was given in the previous tutorial.

Export

To export the mesh and convert it to an external file, one just needs to select it and click with the right button in the viewport. In the 3ds Max Quad Menu it is necessary to select the option *VRay mesh export*. This way the following chart appears. If an object is selected, a message appears in the viewport advising the missed selection of a geometry to export.



■ **Figure 7.16**
Another way of making the export chart appear is writing the following line within MAXScript: ***doVRayMeshExport()***.

Folder - it specifies the path for saving the *Proxy .vrmesh* file(s). As we will show soon, it is possible to create several *.vrmesh* files with the same command.

Export as single file - by using this option, all the selected objects are going to be exported into one *.vrmesh*. Once the meshes are exported and successively imported by means of the *VRayProxy* object, it is impossible to access any of the sub-objects, like vertexes, edges or faces. Furthermore, the exported *.vrmesh* contains information concerning the materials. For this reason it is preferable to divide the original meshes using ID Materials. This way, once imported into the scene as a *Proxy*, the application of the Multi/sub-object appears without mistakes. In case the original mesh is not aligned to the global coordinates [0, 0, 0], but positioned in any point of the 3D space, the Pivot of the *.vrmesh* is placed in the coordinates [0, 0, 0], regardless of the position of the original mesh.

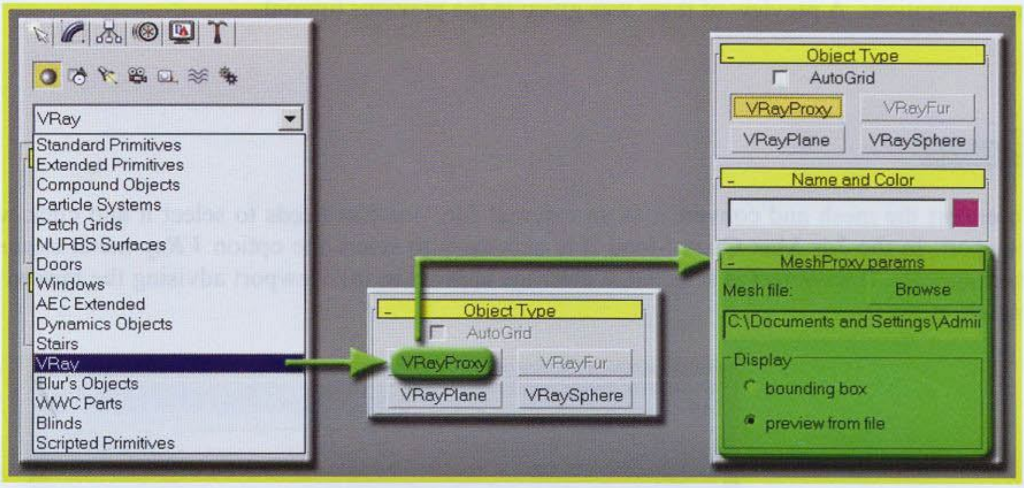
Export as multiple files - with this option, all the selected objects are exported, each one to a single *.vrmesh*. Furthermore, the Pivot of *.vrmesh* is always placed at the centre of the original mesh, on the supporting plane, independently from its position in the mesh before the export.

Automatically create proxy - with this option it is possible to generate, selecting more than one mesh as well, multiple *.vrmesh* files, which are automatically placed inside the scene by *VRay*, in the same position as the original meshes, which are then deleted after the introduction of the *VRayProxy*.

Import

The importation of ***.vrmesh*** is carried out with the use of ***VRayProxy*** objects. It is possible to find them in the pull-down menu Create under ***VRay***, together with three other types of ***VRay objects***.

■ **Figure 7.17**
The ***VRayProxy*** object contains two parameters only.



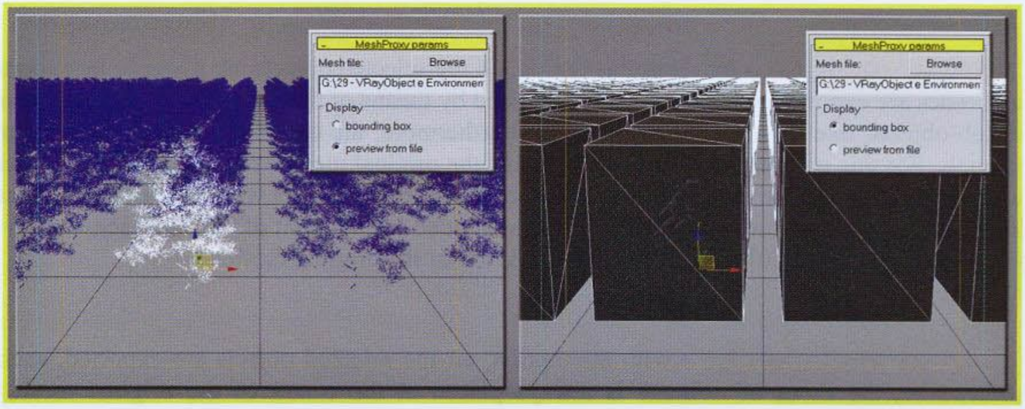
By selecting the ***VRayProxy*** and clicking in the viewport, a panel for selecting the ***.vrmesh*** appears. Once selected, a preview of the geometry appears. With the option ***bounding box*** it is possible to visualize the ***VRayProxy*** by means of a box which has a size corresponding to the volume occupied by the original geometry.

Mesh file - it allows one to manually select the ***.vrmesh*** file.

Display - it controls the way the ***VRayProxies*** are visualized in the viewport.

. bounding box - the ***VRayProxy*** is visualized as a box.
. preview from file - the ***VRayProxy*** is visualized thanks to a geometry simplified by the original mesh. This information is stored directly in the ***.vrmesh*** file.

■ **Figure 7.18**
The forest of 400 trees in the two modes, ***preview*** and ***bounding box***.



NOTES

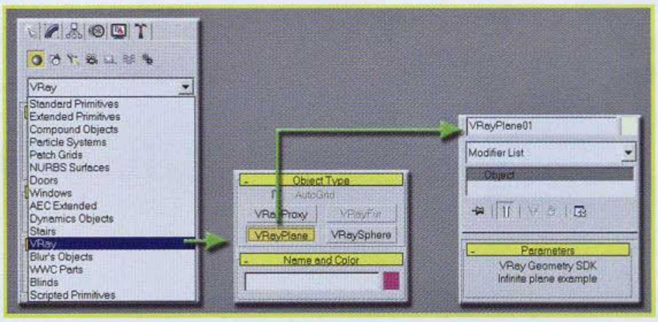
These objects have some limits:

- The ***.vrmesh*** cannot be modified or edited in any way. Any modifier applied is ignored.
- The best solution, if one wants to create many copies of a ***VRayProxy***, is to use instances.
- Materials are not saved within the ***.vrmesh***. Once the ***VRayProxy*** is inside the scene, it is necessary to apply the chosen material manually.
- It is not possible to use the ***Shadows map*** with the ***VRayProxy***, but the ***VRayShadow*** only.
- Problems of ***Motion blur*** may occur if animated (problem solved with the ***1.5 FINAL***).

VRayPlane

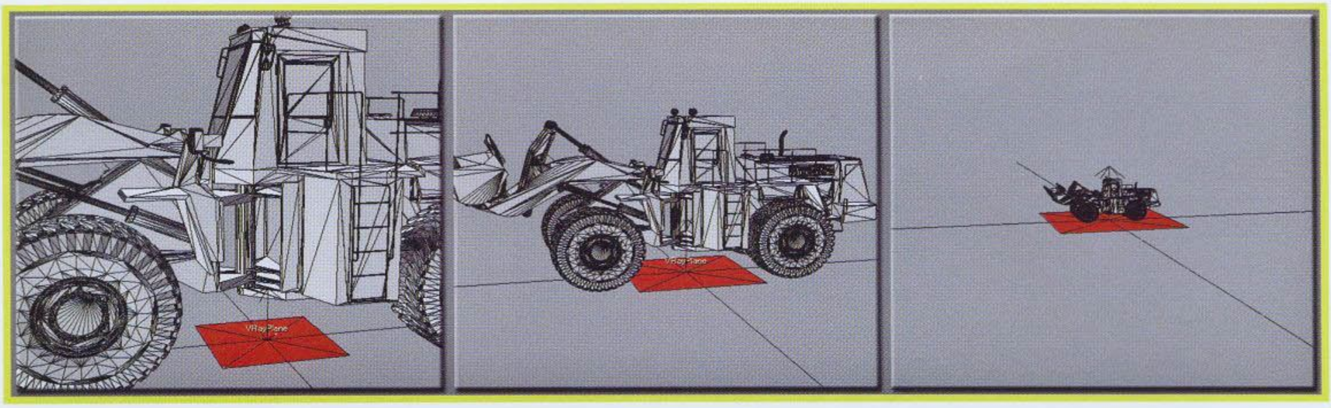
INTRODUCTION

The ***VRayPlane*** is a particular object: it allows one to create an infinite-sized procedural plane. For its creation one just needs to select the ***VRayPlane*** from the menu of the objects and click in the viewport to locate the plane.



■ **Figure 7.19**
The three passages for the creation of a ***VRayPlane***.

Some may ask why not use a normal, very wide geometrical Plane instad of the ***VRayPlane***. This system would be very uncomfortable and less practical, because the plane would physically fill a wide portion of the 3D scene, while the objects would be very little compared to it. Instead, the ***VRayPlane*** is represented by a rectangular icon which has a fixed size, similarly to what happens with the icons of lights or cameras. When approaching them, size remains the same, allowing an easy visual management of the scene.



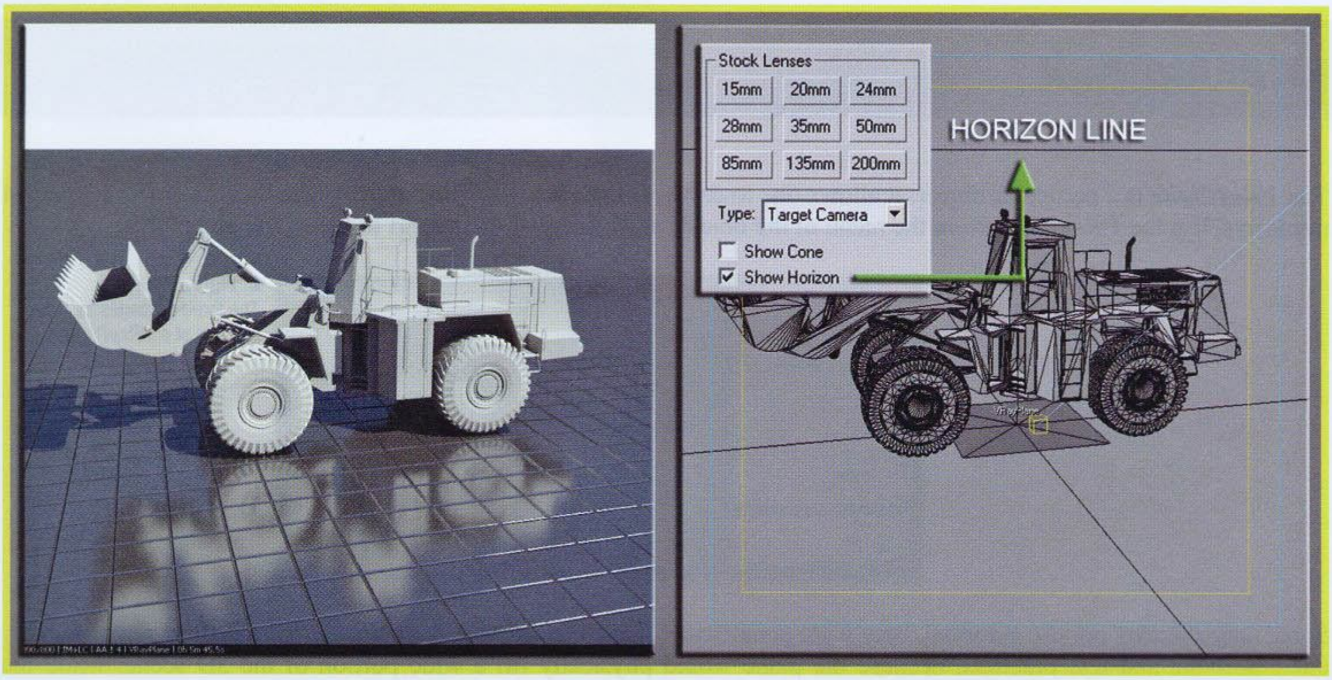
■ **Figure 7.20**
The size of the ***VRayPlane*** in the viewport is the same, both approaching it or moving away from it.

PARAMETERS

This object does not have parameters. It is enough to insert it in any point of the scene, rotate it or move it, and the ***VRayPlane*** generates an infinite plane in length and width. It can also be textured. Some exceptions exist like, for example, the impossibility of applying the ***VRayDisplacementMod*** or other modifiers.

■ **Figure 7.21**
The ***VRayPlane*** does not have parameters.

In the following example a ***glossy*** material has been applied to a ***VRayPlane*** placed horizontally, with a Tile map within the Bump channel. The render does not present any problem. To be noticed how the ***Show Horizon*** parameter of the camera, an option allowing one to visualize the edge of the horizon in the viewport, corresponds exactly to the infinite edge of the ***VRayPlane***.

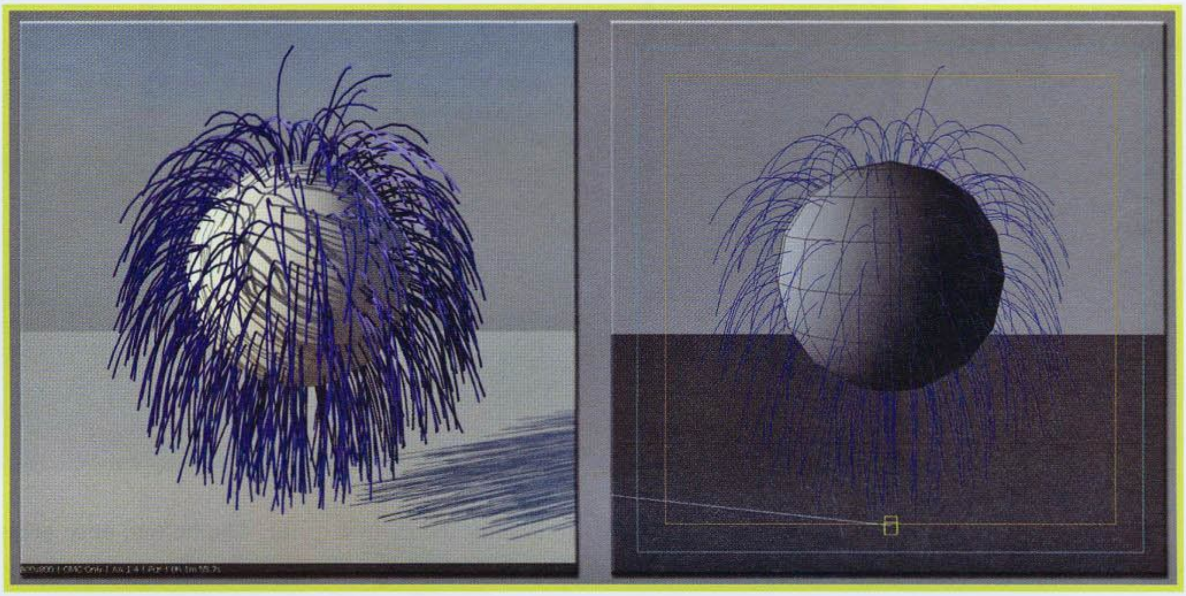


■ **Figure 7.22**
It is possible to determine the horizon line by using the option ***Show Horizon,*** before computing the final render. This feature allows one to identify the line dividing the ***VRayPlane*** from the ***Background*** which, in this case, is completely white. It is possible to locate several ***VRayPlanes*** in the same scene.

VRayFur

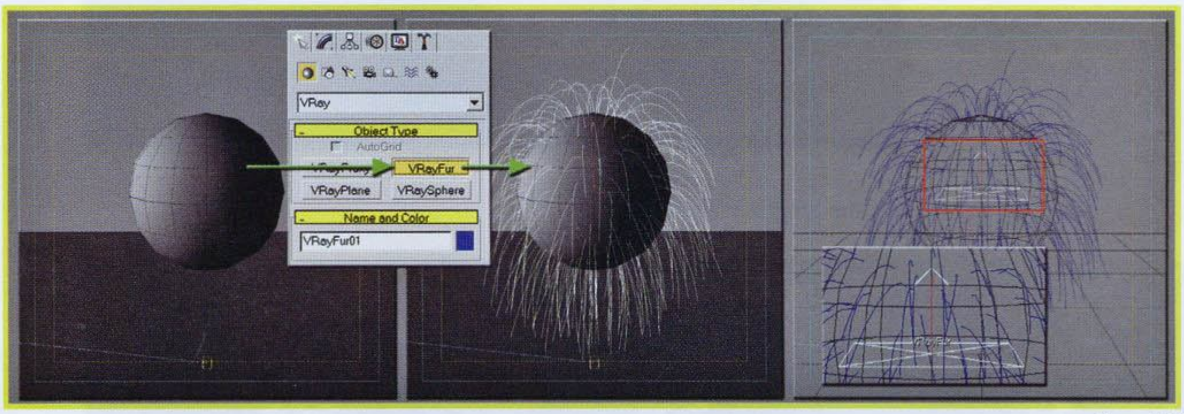
INTRODUCTION

The ***VRayFur*** is an object allowing one to create hair, whorls of grass, filaments, etc… in a procedural way. Therefore, it is not necessary to manually create geometries, such as renderable spline or geometric solutions, but it is enough to "connect" the ***VRayFur*** to a geometry in such a way to make filaments "sprout" from the mesh. The user has many parameters available for modifying filament quantity, the length, etc... .



■ **Figure 7.23**
A simple sphere to which the object ***VRayFur*** has been applied. To be noticed how the fur can be pre-visualized in the viewport.

Unfortunately, this time, the object has lots of parameters, some of which are not immediately comprehensible. But the creation of the fur is instead very simple. The procedure consists of selecting the geometry, in this case a sphere, from which one desires to make the fur "grow". Afterwards one just has to click on the object ***VRayFur*** and an icon appears automatically, aligned to the pivot of the geometrical object, which represents the ***VRayFur***. If any geometry is selected, ***VRayFur*** is unusable and greyed out.

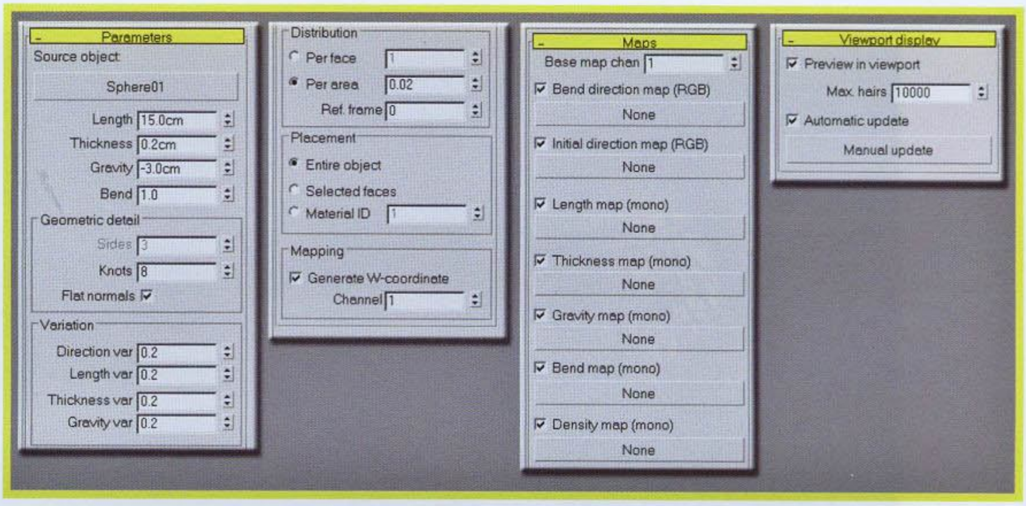


■ **Figure 7.24**
Notice the ***VRayFur*** icon at the centre of the sphere. Its behaviors in the viewport is the same as that of ***VRayPlane***. Getting further away or approaching, the size of the icon in the viewport does not change. Furthermore it can be moved at will by the user. The blue color of the ***VRayFur*** derives from the Wire Color of the ***VRayFur*** icon.

It is possible to move the icon in any case without affecting either the result or the disposition of the filaments on the geometry.

Parameters

The parameters for the regulation of the ***VRayFur*** are many. Not just the numerical values like the ones present in the ***Parameters*** rollout. It is possible to modify the appearance of ***VRayFur*** thanks to maps which can be more or less complicated, by using the options of the ***Maps*** rollout. The last rollout, finally, allows one to manage the preview in the viewport.

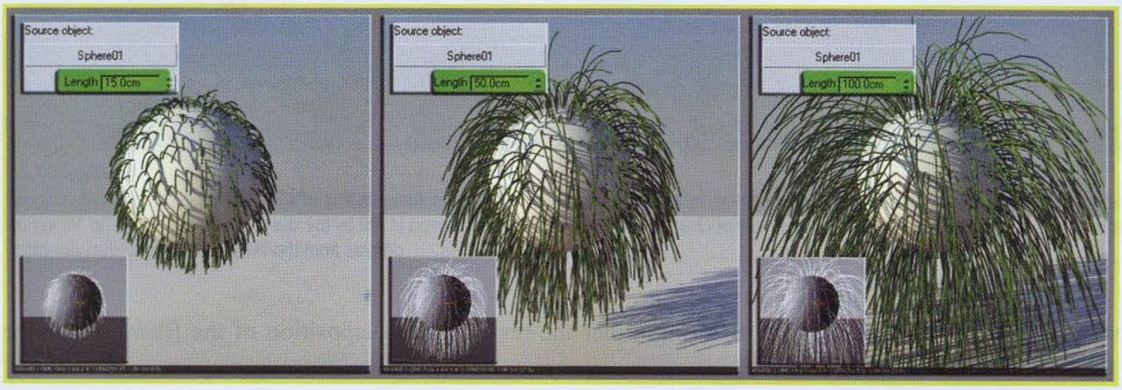


■ **Figure 7.25**
The long list of parameters available for the ***VRayFur*** object.

Source object - clicking on this button it is possible to modify the emitter of fur. The button, once pressed, remains active to allow the selection of a geometry.

■ **Figure 7.26**
Two different geometries used as emitters for the same ***VRayFur*** object.

Length - this parameter sets the length of the filaments.



■ **Figure 7.27**
Some values of length for the ***VRayFur***.

Thickness - this parameter sets the general thickness of the filaments.

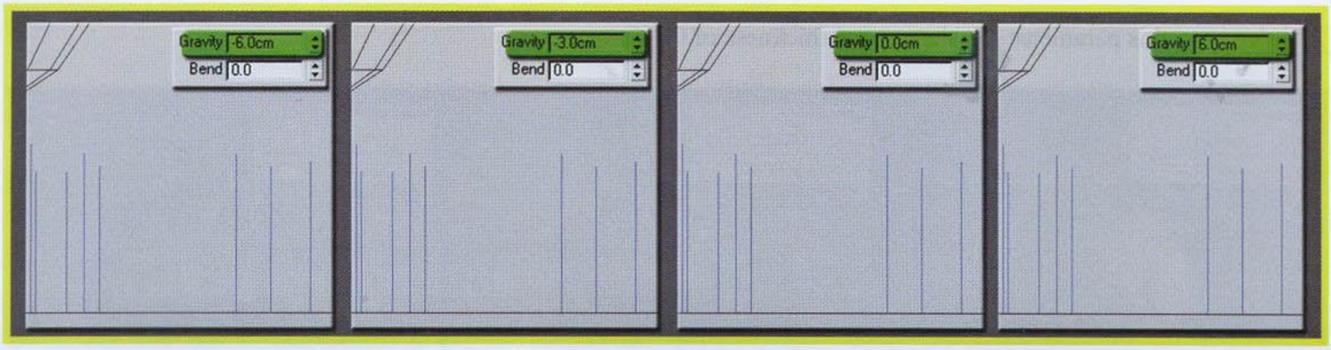
■ Figure 7.28
Different thicknesses of the ***VRayFur***.

Gravity - this parameter controls the general influence of gravity which, working along the global Z-axis, attracts the filaments upwards or down.

■ Figure 7.29
It is possible to defy gravity with the ***Gravity*** parameter! Its effect is similar to that of a magnet attracting filaments towards itself.

With the gravity set to 0.0 hairs does not appear straight, as might be expected. Instead it is slightly bent. This happens because within ***VRayFur*** some parameters are present which randomly insert variations to the direction, the length, the thickness, etc… In this particular case, since the option ***Direction var*** is active, that is the option regulating the direction of the hair, the fur is not totally straight, despite the gravity being 0.0. Another parameter exists, ***Bend***, which controls the bending of ***VRayFur***. The latter operates in a very similar way to the ***Gravity*** parameter, but nonetheless different.

Bend - this parameter controls the general bending of the filaments. ***Bend*** operates together with the ***Gravity*** parameter. To better understand their functioning, some screenshots, directly derived from the viewport, are shown. For the occasion the emitter has been modified, and a geometrical plane is used instead in order to make the difference between the parameters ***Gravity*** and ***Bend*** clearer.
First of all the ***Bend*** parameter is turned off, and one changes ***Gravity*** only.



■ **Figure 7.30**
With the ***Bend*** option deactivated, the ***Gravity*** effect does not have any influence.

When no bending is active, that is, filaments cannot be bent, the filament becomes rigid and the gravity, both positive or negative, does not affect ***VRayFur***.

In the following, the two operators are inverted: ***Gravity*** is turned off, operating on the ***Bend*** only.

■ **Figure 7.31**
The Bend effect affects the ***VRayFur***, by turning ***Gravity*** off.

In this case, even though gravity is absent, the ***Bend*** parameter influences the appearance of ***VRayFur***. With value 0.0 filaments are completely straight, and the higher ***Bend*** value is set, the more the filaments bend.

Now the ***Bend*** is set to 1.0, and ***Gravity*** is changed.

■ **Figure 7.32**
The gravitational attraction influences the bending of the ***VRayFur***.

By changing the ***Gravity*** option with ***Bend*** set to 1.0, the filaments do not bend as they had when only the ***Bend*** parameter was active, but they form a very smooth curve. This way the nearer segments to the plane are not squeezed, but they are almost perfectly vertical.

Sides - this parameter, which allows one to modify the number of sides each filament is composed of, is currently disabled and set to three sides. This way the section of the filament is triangular. To solve the problem of the low number of segments, *VRay* automatically shows the normal of the faces perpendicularly to the camera. Furthermore, the three sides are smoothed, as would happen if the Smooth modifier were applied to them. Therefore this limitation is not all that important for medium/little size filaments.

Knots - this parameter controls the number of segments the single filament is composed.

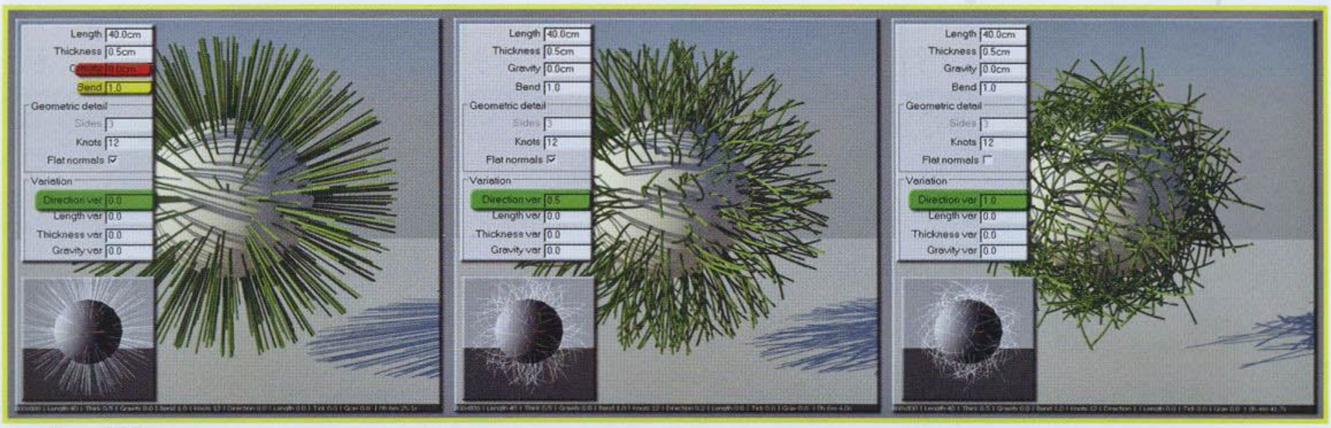
■ **Figure 7.33**
The number of segments, other than the quality of the filaments, affects their bending.

The image clearly shows the meaning of the ***Knots*** parameter. A filament corresponds to a line. A straight line contains two vertexes, one beginning the line and the other finishing it. If one tries to modify the hair, it will not bend, due to the low number of segments. By increasing the number of segments quality increases as well as rendering time.

Flat normals - if this option is active, which happens by default, the normal of the geometries the ***VRayFur*** is composed of is always facing the camera, giving the idea of hair formed by sharp-edged polygons. Instead, by deactivating it, hair appears as a cylinder with a variable smoothing along the whole filament. In the first case, perhaps the less realistic one, the absence of smoothing may help the ***antialiasing***, making the sampling easier. Furthermore, the visibility of smoothing depends on the section of the ***VRayFur***. When filaments are too thick the activation or deactivation of the ***Flat normal*** parameter is negligible in the render.

■ **Figure 7.34**
In this example rendering times are the same. Notice the slight smoothing of the filament with the ***Flat normal*** parameter deactivated.

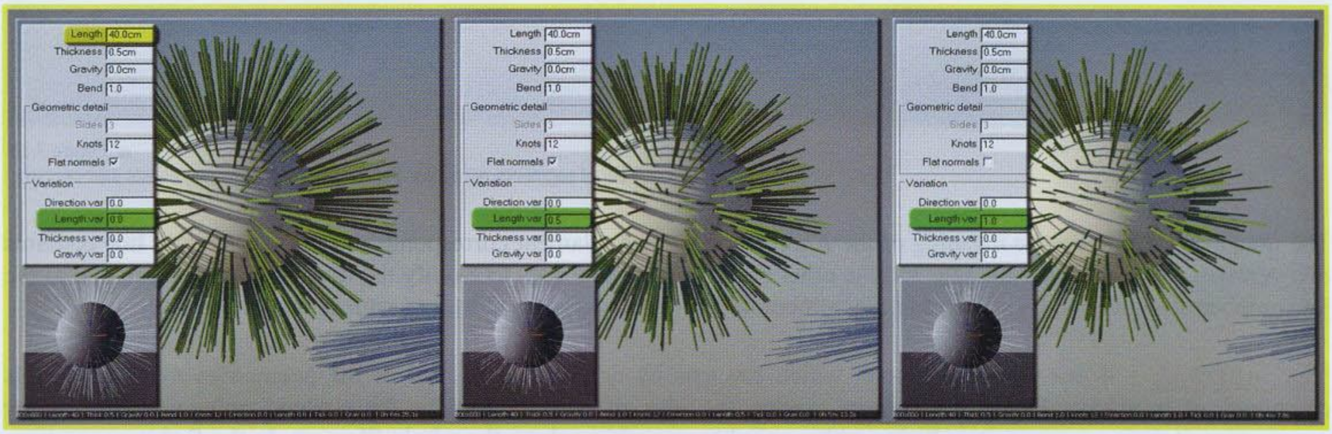
Direction variation - this parameter controls the global variation of the direction of all the filaments on the emitter. ***Direction variation*** affects the behaviour of both ***Gravity*** and ***Bend***.



■ **Figure 7.35**
The action range of the ***Direction var*** ranges from 0.0 to 1,000,000 maximum.

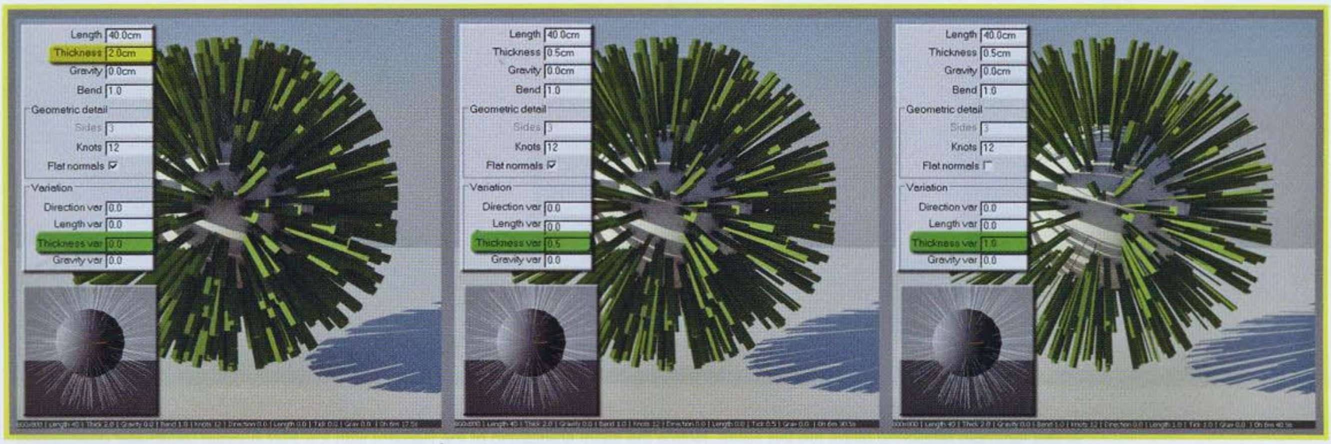
The gravity effect has been turned off in the previous example. Only bending has been left active, with a value of 1.0. With the ***Direction variation*** parameter set to 0.0, filaments are completely straight. By increasing the value, the ***VRayFur*** starts to bend under the action of the ***Bend*** parameter.

Length variation - this parameter controls the global variation of the length of all the filaments present on the emitter.



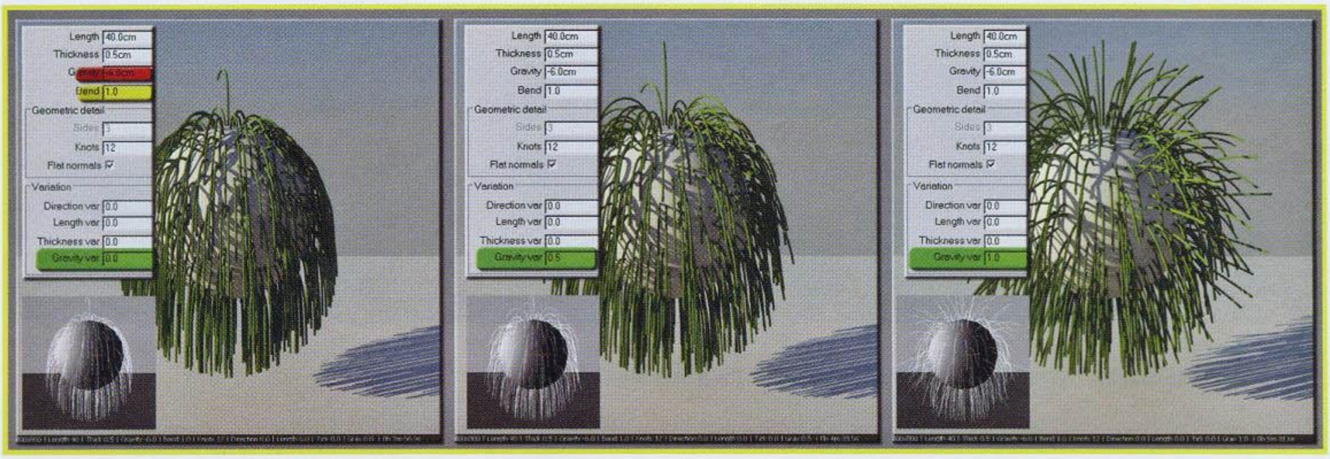
■ **Figure 7.36**
Length variation: another parameter for improving the realistic effect of filaments.

Thickness variation - this parameter controls the global variation of the thickness of all the filaments of the ***VRayFur*** object.



■ **Figure 7.37**
In the example, the section of the filaments spans from 0.0cm and 2cm with a value of ***Thickness var*** = 1.0.

Gravity variation - this parameter controls the global variation of the influence of the gravity on all the filaments of the ***VRayFur*** objects.

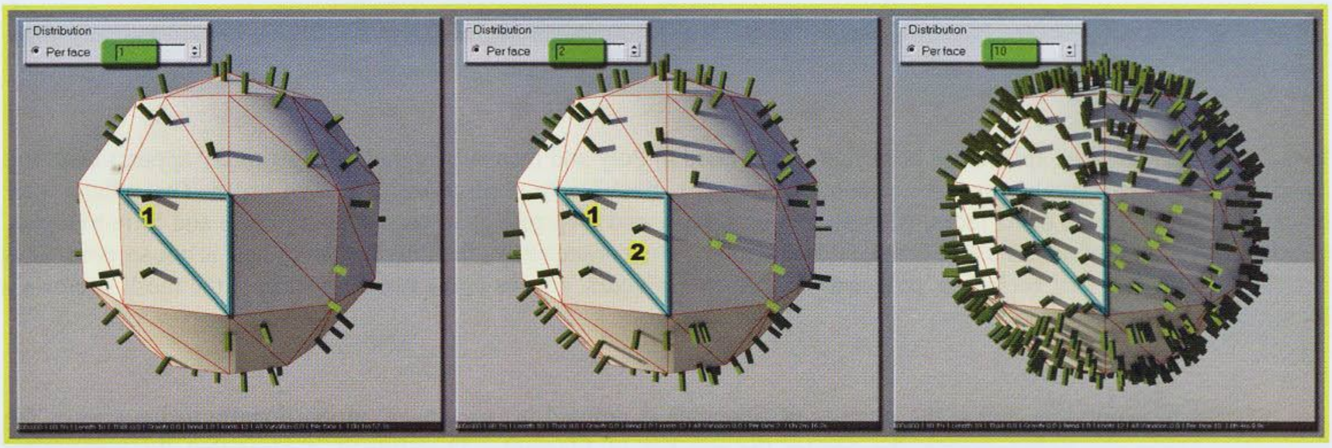


■ **Figure 7.38**
The amount of random variation of the ***Gravity*** parameter lowers the attraction of the filaments, in this case, downwards.

With the value ***Gravity variation*** set to 0.0, the attraction due to the ***Gravity*** parameter set to -6.0 is maximum. The more the variation increases, the lower the influence of the gravity is, and the ***VRayFur*** is affected less by the downward attraction.

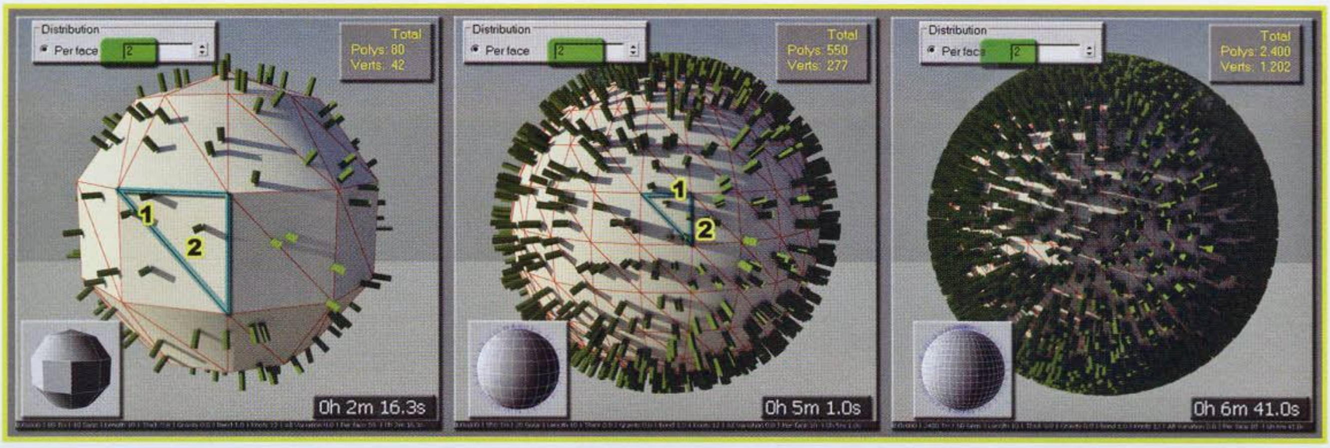
Distribution - the parameters contained in the ***Distribution*** section set the density of the filaments. In this case it sets the amount of filaments emitted by the polygonal sphere. It is possible to select two kinds of distribution: per faces and per areas.

Per face - by using this method, the number of filaments is computed per each single face of the geometry. It is to be noticed that each single triangle, composed by three sides, is meant as "face". For example, a value 2 means two filaments are emitted from each triangle. The sphere is reduced to few triangles in order to show the functioning of this parameter: exactly 80. This improves the visibility of each triangle.



■ **Figure 7.39**
Each triangle contains the same number of filaments set by the ***Per face*** parameter.

The arrangement of hair is random, and each time their number increases, the position of the existing ones keeps constant. There is an incremental increase in the hair, adding filaments to those already existing.
What happens when, the value set in the ***Per face*** parameter being equal, the resolution of the mesh changes, that is when the number of polygons inside the model increases? Quite simply, the sphere is going to have more filaments, in a proportional way. In this case, with ***Per face*** = 2, with a single face there are two filaments, with two faces four, with four triangles eight and so on. To know the exact number of filaments one just needs to multiply the number of faces the sphere is composed of to the value of the ***Per face*** parameter.



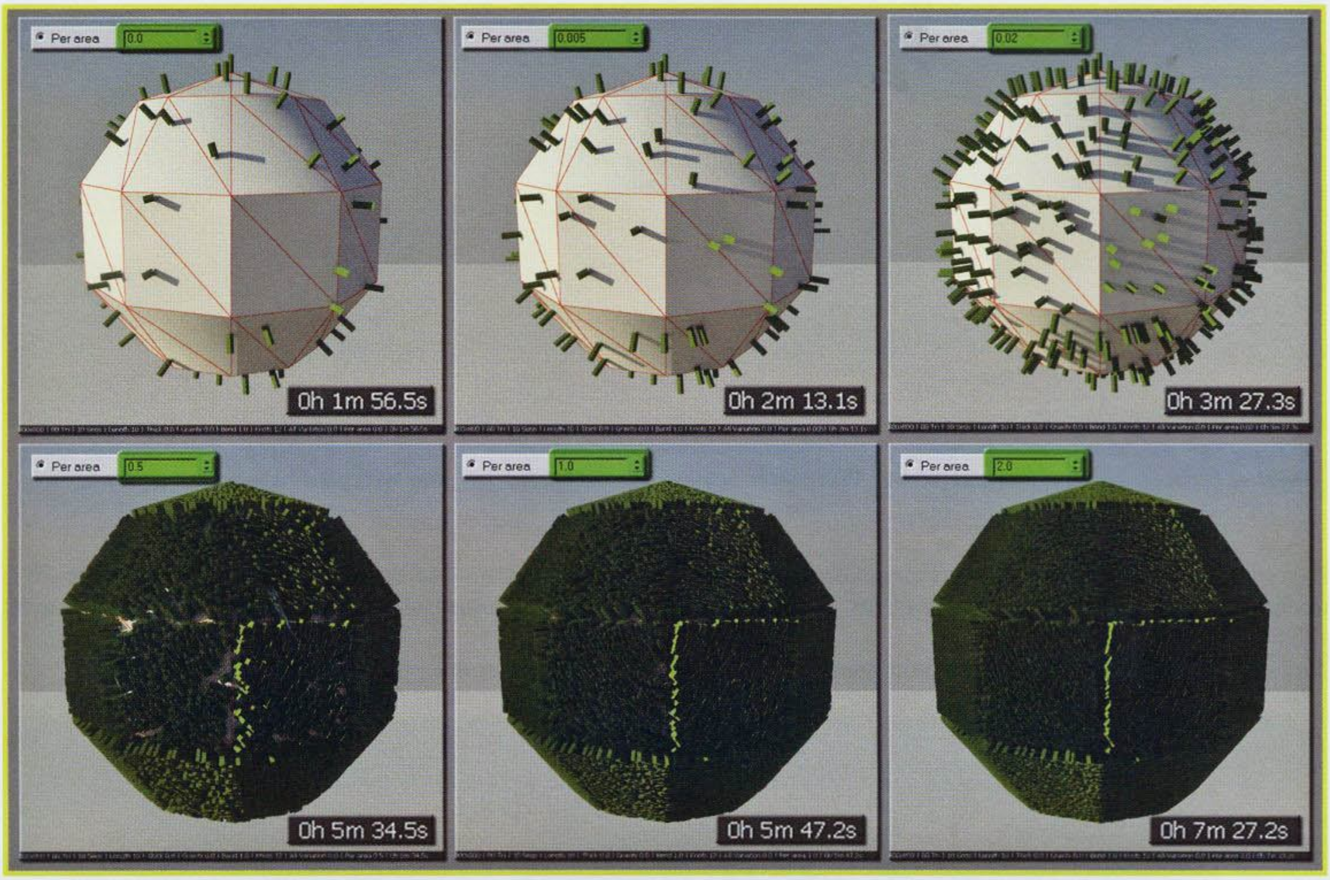
■ **Figure 7.40**
The mesh is more complex. This happens when there is an increase in the density of the ***VRayFur***.

One last observation must be done. The mesh used until now is regular, with polygons and faces of the same sizes. What happens if the triangles have different shapes?

■ **Figure 7.41**
Mesh with a complex polygonal resolution.

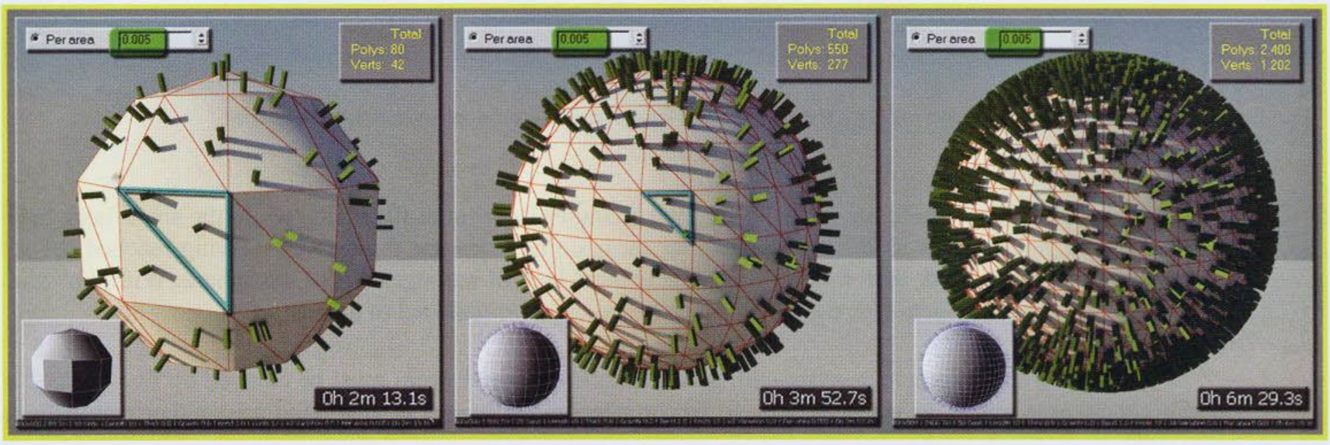
In the example above the sphere has been firstly converted in Poly and then subdivided. The geometry appears tidy. But by making the edges hidden by the geometry appear, it is possible to observe some lack of order. The sphere, after having subdivided some of its faces, is composed by narrow and long triangles, very wide triangles and little triangles. In this case the ***VRayFur*** has a less foreseeable situation. As a consequence, the distribution of hair is not so even. This is caused by the fact that each face, independently on its wideness, always emits two filaments: if the mesh is dense a lot of hair is produced.

Per area - by using this option the number of filaments is defined in percentage, proportionally to the size of the faces. The bigger the polygons are, the greater the number of filaments. Therefore it is clear that, in a geometry containing variable size polygons, the biggest faces are going to have more filaments compared to the little ones. The difference with the ***Per Face*** parameter is exactly this: with ***Per Face***, every face always has the same number of filaments, regardless of the size of the face.



■ **Figure 7.42**
Increasing the number of filaments, rendering time increases as well.

As in the previous example, in this case too, the density of filaments increases as the 3D model's geometric complexity increases.



■ **Figure 7.43**
The more the geometry is complex, the greater the number of filaments.

As far as the geometries with variable triangulation are concerned, the behaviour of the ***Per area*** parameter is a little particular. In the previous case it was very simple to determine the amount of filaments. In fact it was enough to set the number of hair per triangle and the result was foreseeable. The smaller the triangles of the mesh, the more the surface would have been dense with filaments. Therefore, fur could be strongly lacking in homogeneity in different areas. By increasing the parameter ***Per face*** this lack of homogeneity would not be corrected in any way. For all the faces, big or little, the number of filaments would constantly increase.

■ **Figure 7.44**
Use of the ***Per face*** parameter on a complex geometry. The red arrow shows the areas with higher density, while the blue one the ones with lower density.

Every time the ***Per face*** parameter is increased, the number of filaments grows on the sphere. In this case a value 50 is not enough to fill the largest triangle of fur. On the other hand the other geometries are very dense, especially the most subdivided ones.
Now observe a similar situation with the same sphere, but with use of the ***Per area*** parameter.

■ **Figure 7.45**
Employment of high values of the ***Per area*** parameter on a complex geometry. The red arrows show the areas with a higher subdivision.

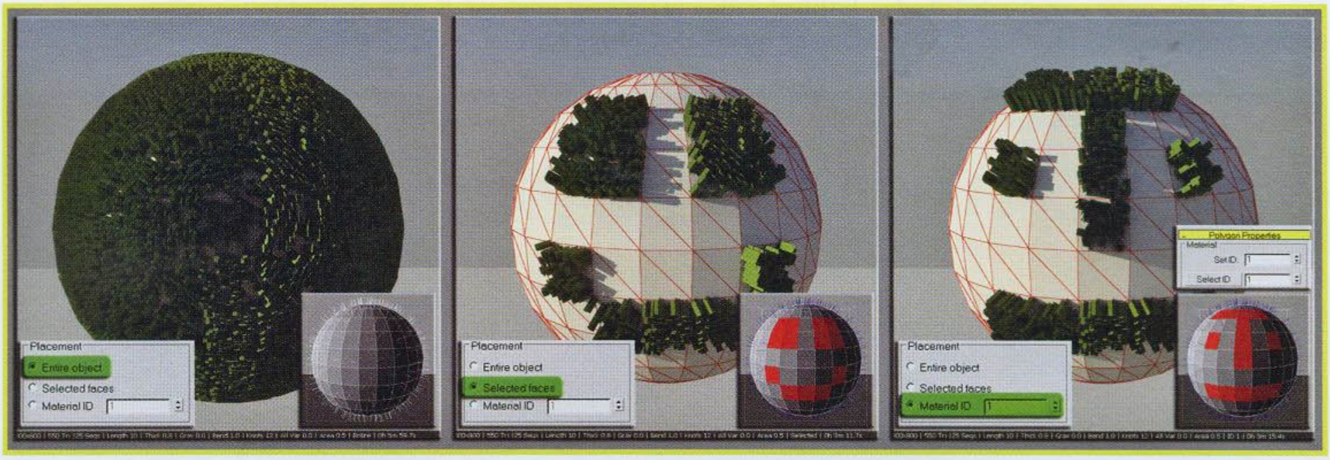
In the test on the left, the result is quite obvious. The biggest triangles have more filaments, while the small ones have less. Now the ***Per area*** parameter is modified. The smaller faces do not increase the density. Having reached the set coverage, no other filament is added, therefore reducing the general lack of homogeneity of the sphere. With the value 1.0, the sphere is completely homogeneous, even though the triangles it is composed of are not so.

Placement - this section allows one to specify, with greater precision, which areas must be covered by filaments. It is possible to select the whole object, as we have done so far, or select the polygons in a targeted way, either manually or via the material ID.

Entire Object - all the faces composing the geometry generate the filaments.

Selected face - only the selected faces emit filaments.

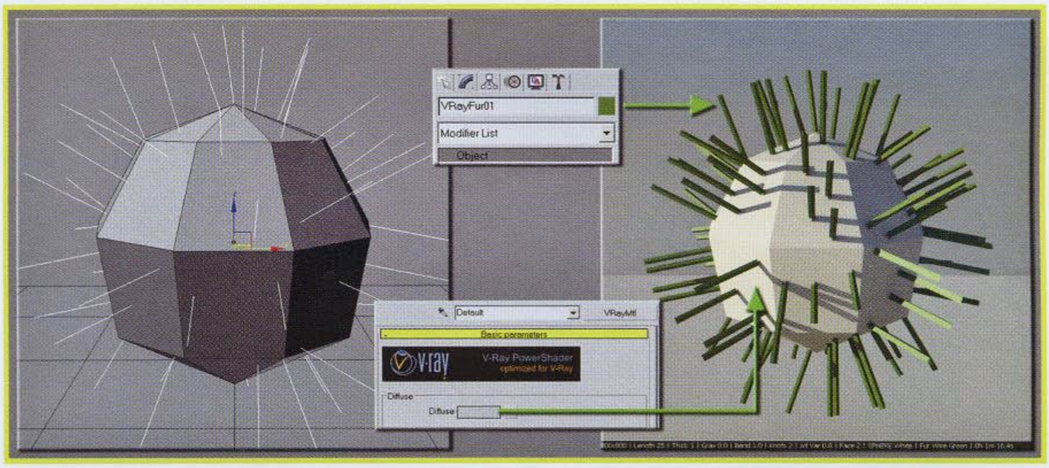
Material ID - with this parameter it is possible to generate filaments specifying the number of face IDs and therefore the polygons which have to generate the fur.



■ **Figure 7.46**
Whether polygons are selected or the ***Material ID*** option is selected, in the viewport it is not possible to observe the correct position of the filaments.

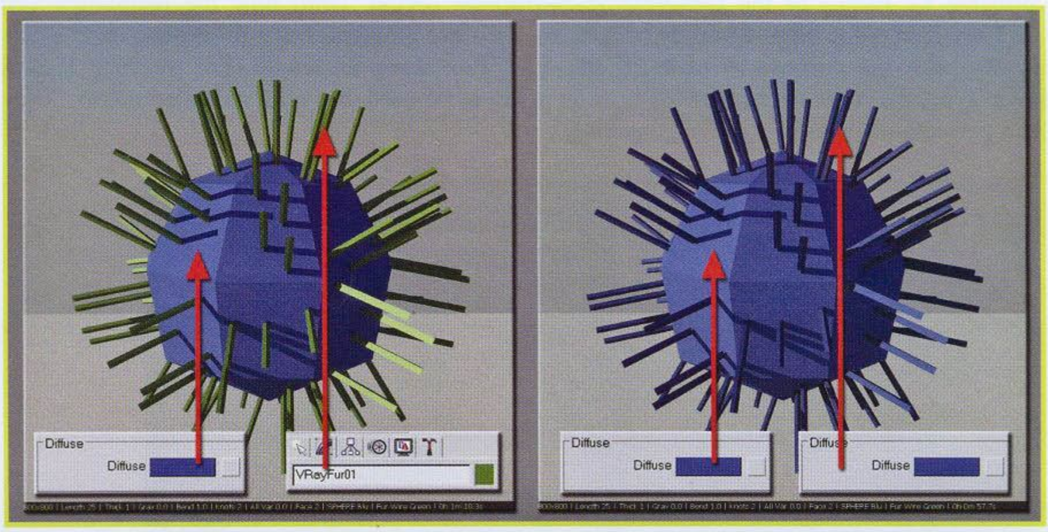
Mapping - the parameters found in this section manage the mapping of single filaments. It is necessary to give information which allows one to understand how ***VRay*** textures its filaments.

The materials used so far to render the various tests are simple. The sphere and the plane have a white color, while for the filaments the color is defined simply by the Wire Color of the ***VRayFur*** icon.



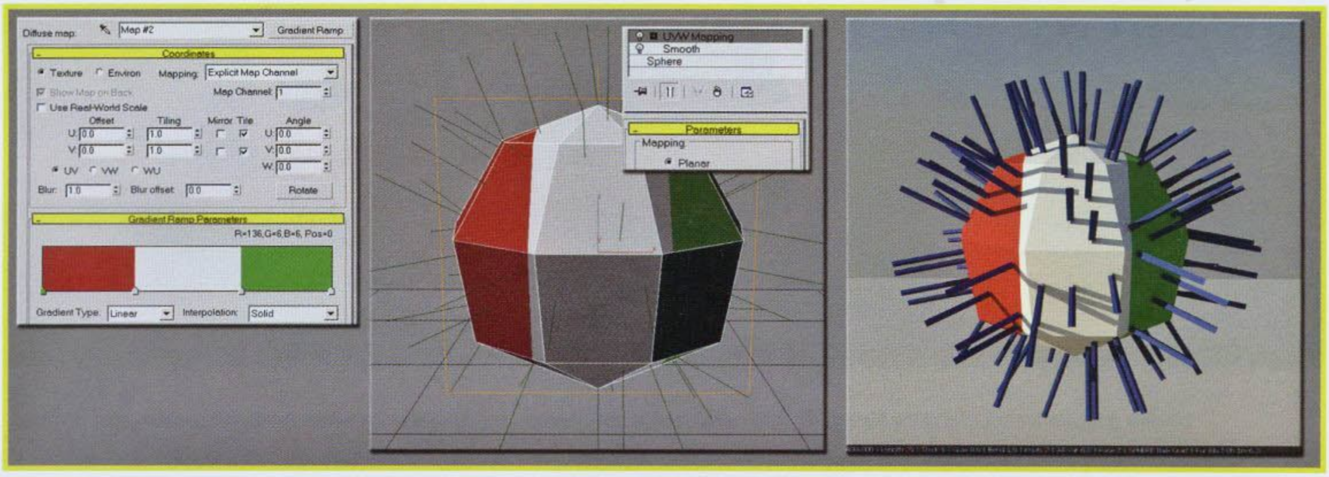
■ **Figure 7.47**
In the tests carried out so far no mapping has been used. A light grey color has been applied to the sphere, while no material has been assigned to the ***VRayFur***. Notice that its Wire Color is green, as in the render.

Now one might ask how to color, map, and texture the filaments. First of all a blue material is assigned to the sphere, keeping the ***VRayFur*** Wire Color unchanged. Assigning afterwards the same ***VRayMtl*** blue material to the filaments as well, both the objects have the same material.



■ **Figure 7.48**
It is possible to manage the shading of the sphere and of ***VRayFur*** with one material only.

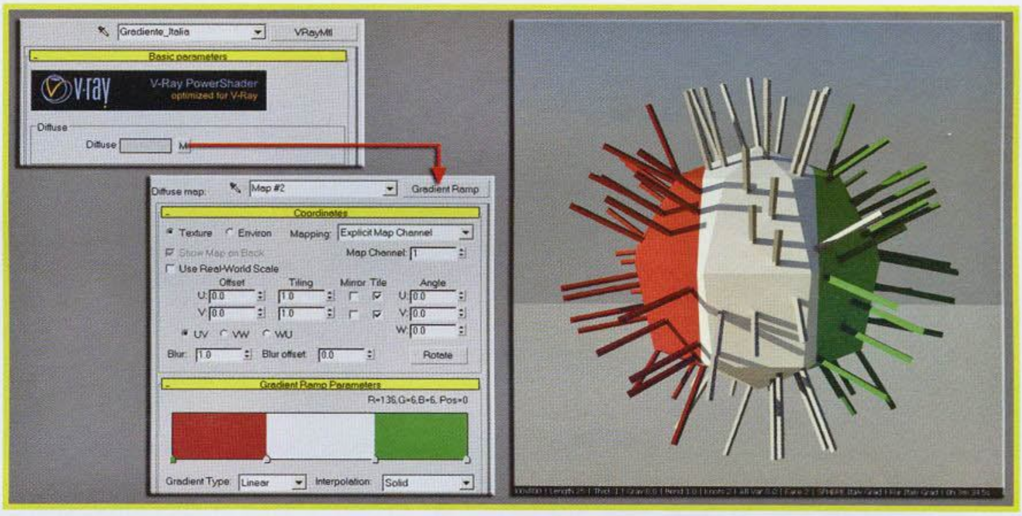
Now a slightly more complicated shader, fulfilled with a procedural gradient, is now applied to the sphere, which was mapped previously by means of a UVW Mapping modifier of planar parallel to the view kind.



■ **Figure 7.49**
The sphere only is mapped with a procedural texture. The ***VRayFur*** remains textured with the previous blue color.

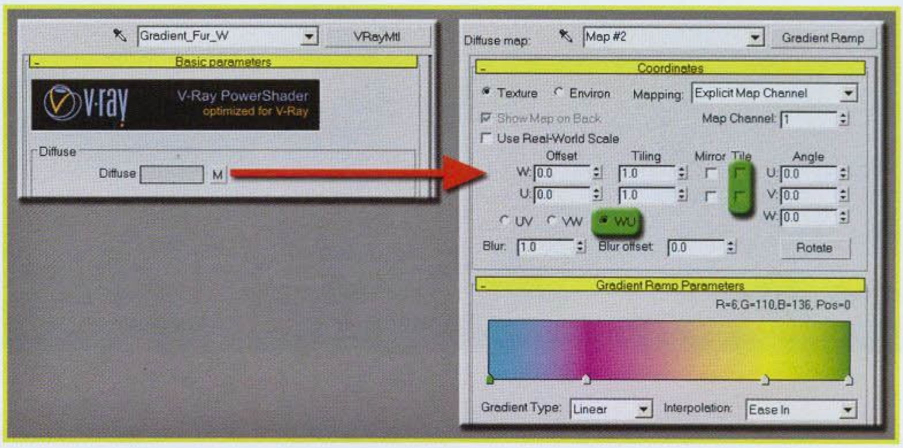
The sphere correctly reports the mapping coordinates and the procedural gradient is clearly visible. In the following the same shader is going to be applied to the ***VRayFur***.

■ **Figure 7.50**
The ***VRayFur*** and the emitter have the same material, but also the same mapping coordinates.



VRayFur acquires the mapping coordinates of the object to which it has been applied, and for this reason the filaments are textured as the sphere is. But is there a method for mapping the filaments longitudinally, along their extension? The answer is yes, and it is possible by means of the ***Generate W-coordinate*** option. Until now the mapping's UV coordinates have been used. They have allowed to correctly texture the tricolour gradient, but a third dimension exists as well, W, which, in this case, has the purpose of mapping the ***VRayFur*** along its length. For the next tests a procedural map of the gradient type, the same as the previous one is going to be used, but with modified colors. They are going to clarify and give a correct insight as to what is happening.

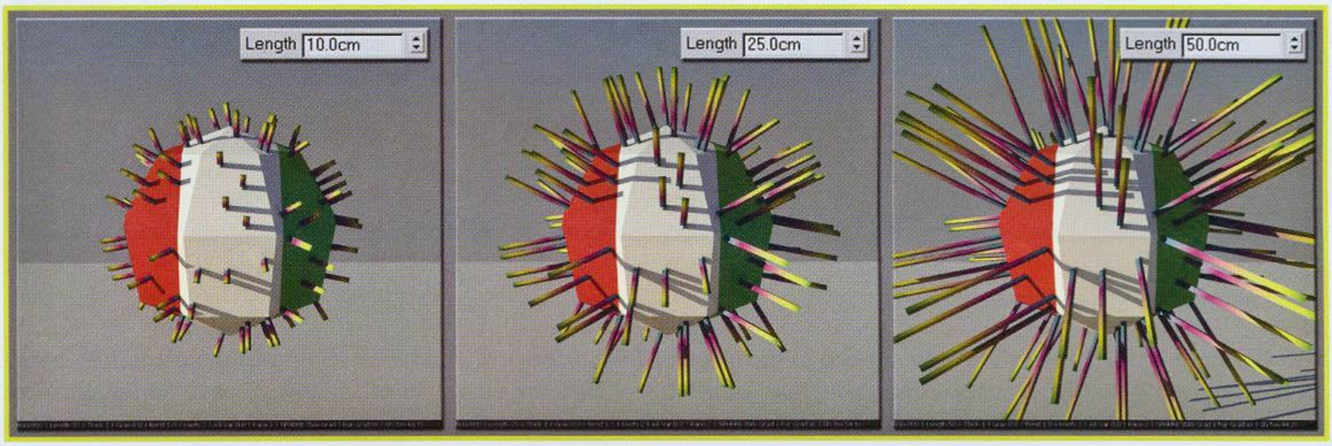
■ **Figure 7.51**
The procedural texture with a few modified parameters.



To correctly texture the filaments, some parameters of the Gradient Ramp have been modified. In order to deepen knowledge about them we suggest you to read the 3ds Max on line guide.

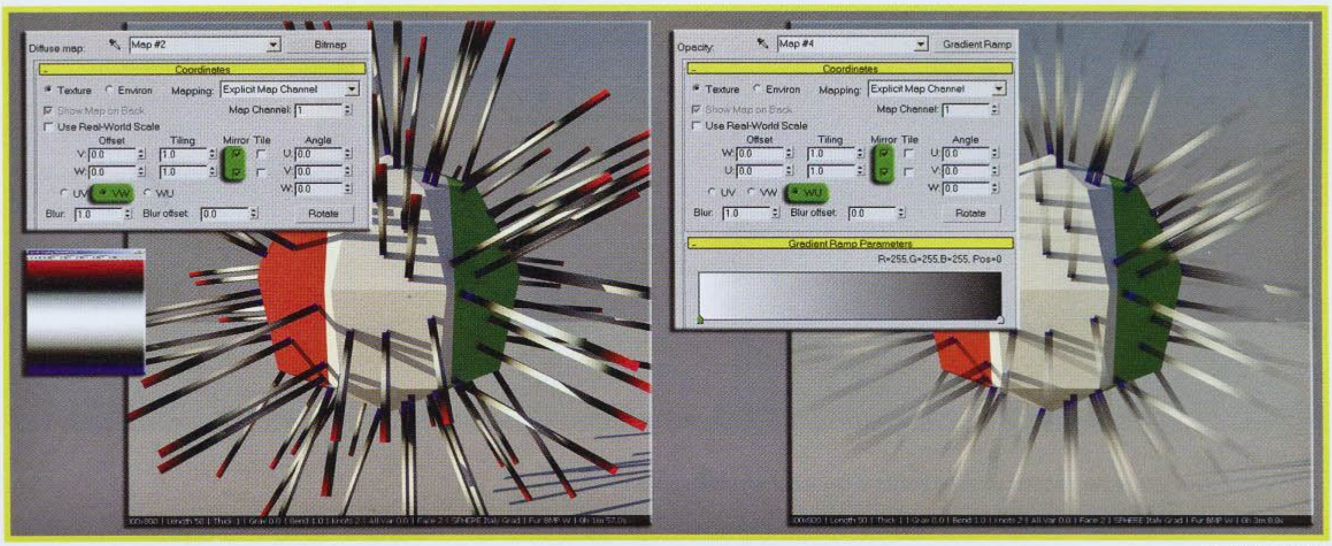
■ **Figure 7.52**
Activation of the ***Generate W-coordinate*** parameter.

By using the ***Generate W-coordinate*** parameter, the gradient is correctly computed along each filament. By modifying the length of the ***VRayFur***, the mapping adapts to the new size. This way the procedural map always covers the entire filament.



■ **Figure 7.53**
By varying fur length, the gradient is always correctly computed.

Also using BitMaps instead of the procedural ones, or the procedurals in the opacity channel, the ***VRayFur*** is correctly rendered.

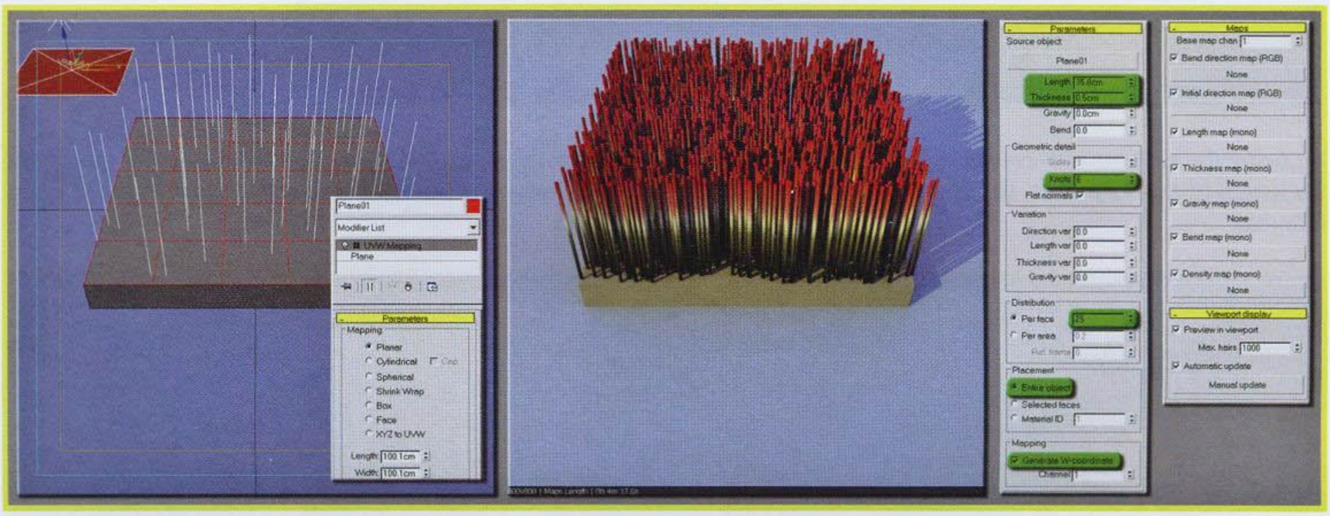


■ **Figure 7.54**
Application of raster textures and use of opacity.

Maps

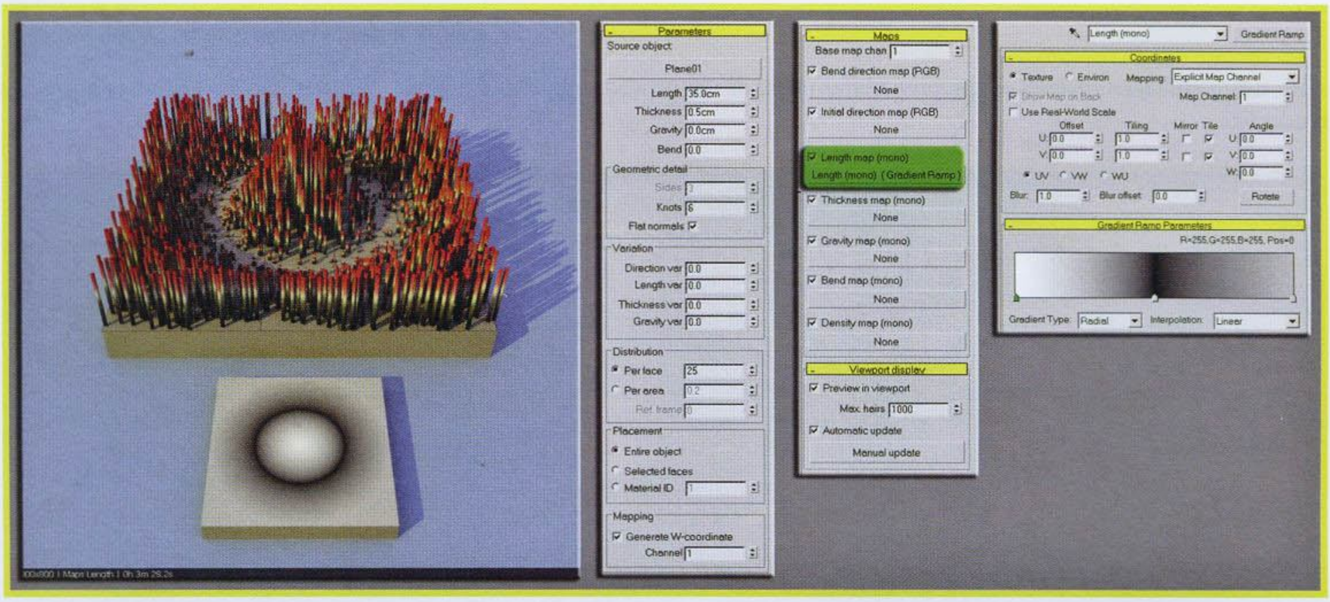
The parameters discussed up to now allow one to operate on the properties of the filaments, such as the length, the thickness, the gravity, the bending, etc… by means of numerical inputs. This has made it possible to recreate numerous effects. But these variations do affect all the filaments of the ***VRayFur*** object in the same way. This means it was possible to set the thickness of the fur, set a variation in order to obtain small, medium and large size sections, but it was not possible, for example, to make the outermost filaments thinner, and the central ones thicker. For this purpose, the Chaosgroup has introduced the maps. Thanks to this stratagem it is possible to accurately modify, by means of the colors of a texture, the value of a specific property, such as the length of the filaments.

To better understand this idea look at the following example. It is a simple plane, mapped in a planar way and used as emitter. Under the plane a box works as a base. The fur is textured with a colored Bitmap with a gradient in order to improve the clarity of the test. The initial settings are very simple. No bending, gravity influence and no kind of variation. Filaments are all of the same length, exactly 35 cm.



■ **Figure 7.55**
Same length filaments applied to a plane.

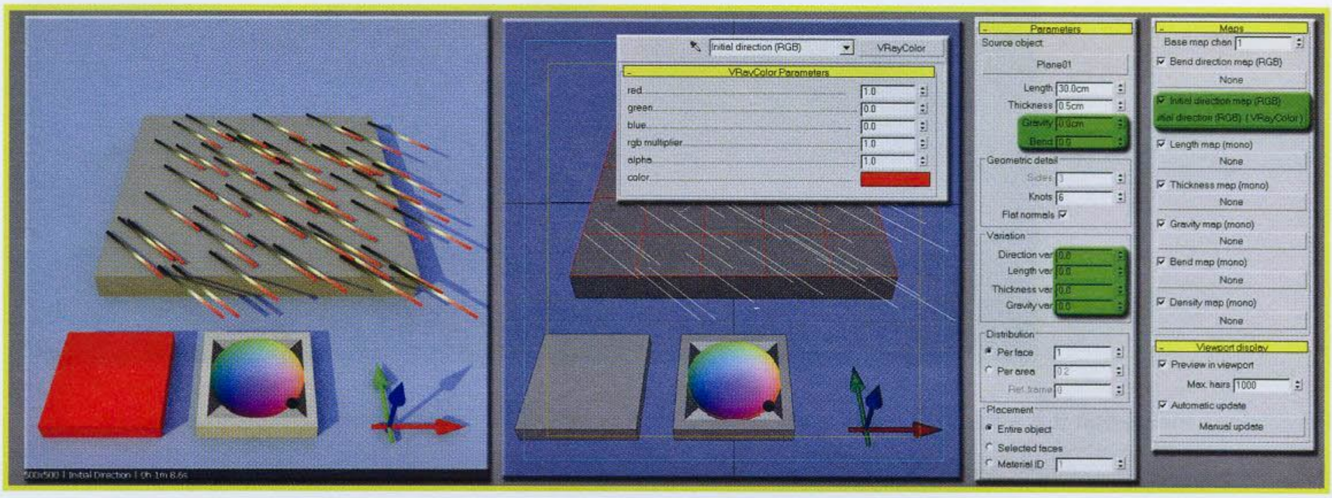
Now by using a black and white map applied to the channel ***Length map***, it is possible, by using the plane's mapping coordinates, to define the exact position and length of the fur. Where the map is black, filaments have no length. Vice versa, where the texture is white, filaments have the highest length possible, that is, in this case, 35 cm.



■ **Figure 7.56**
Variation of the length of the filaments obtained thanks to the use of a procedural map.

The idea explained above, that is, the definition of the length of the filaments by means of a map, is a phenomenon which can be extended to six other properties.

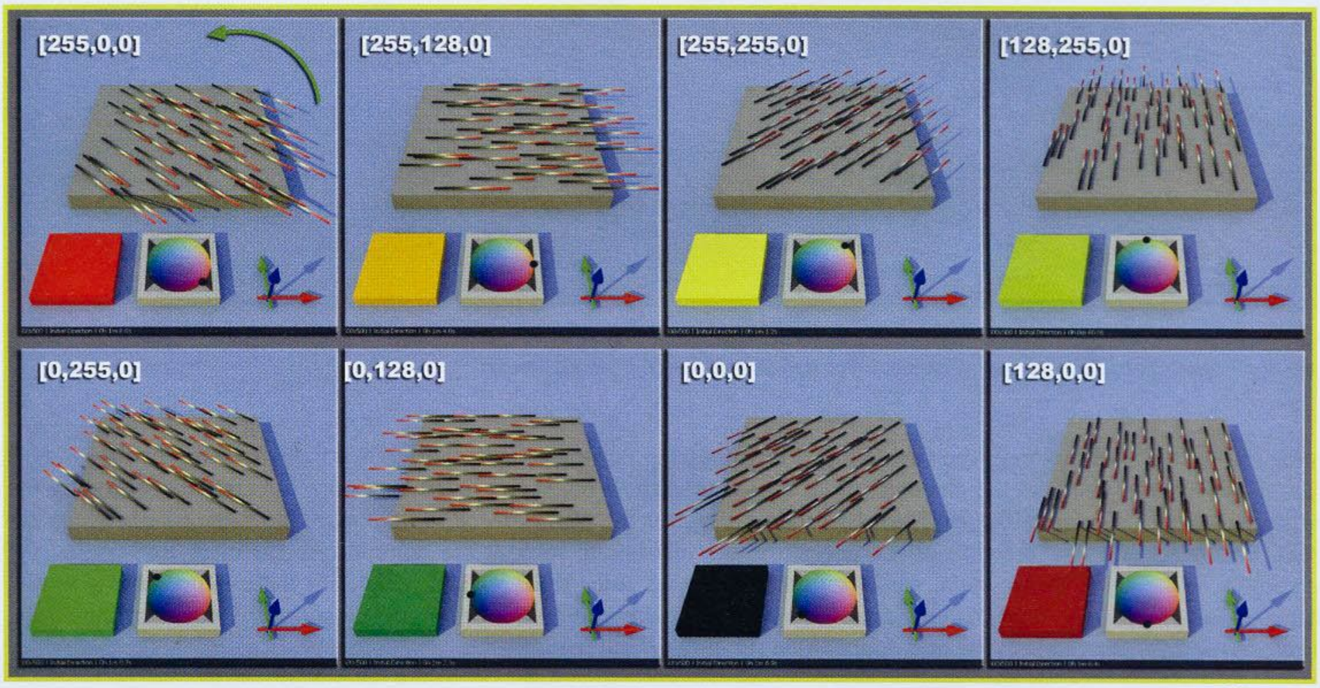
Initial direction map (RGB) - the second map allows one to define the direction of the filaments. In the previous example, dedicated to the ***VRayFur*** length, a black and white map has been used. In this case an RGB map has to be used. The ***VRayColor*** map has been chosen. For the definition of the ***Initial direction color*** parameter, all filaments have the same color and direction. In fact, with both the ***Gravity*** and ***Bend*** parameters turned off, the only way to control the direction is to use a map in the ***Initial direction*** channel.



■ **Figure 7.57**
Completely squeezed filaments.

In the scene some elements are present in order to better understand the test. The box down on the left represents the color expressed by the ***VRayColor*** map, the hue which determines the direction of the filament. The central box, the most important element, is a rack representing the colors related to hair direction. The most saturated colors are arranged in the outermost zone of the circle. The closer one gets to the centre, the brighter the colors, tending towards a cyan color.

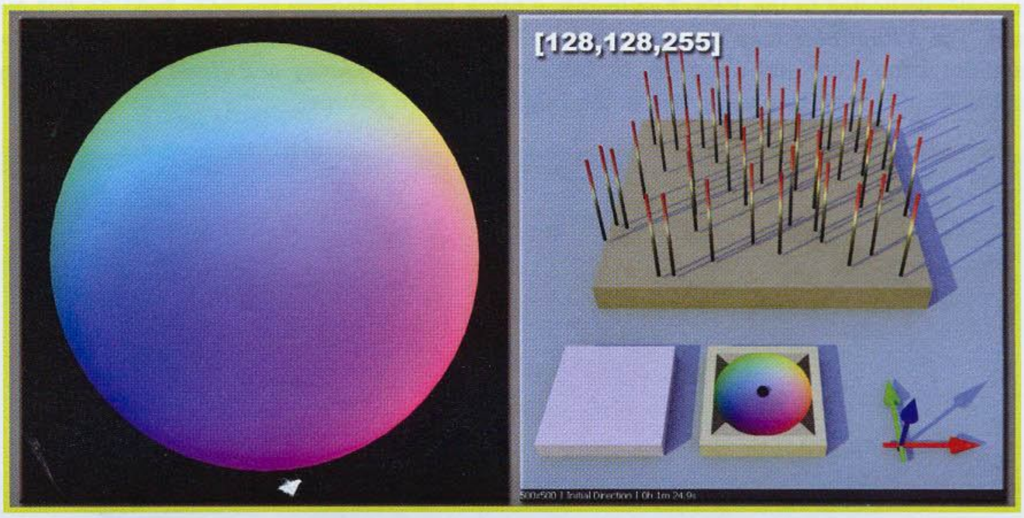
Now pay attention to the following images.



■ **Figure 7.58**
All 360° are covered by RGB combinations.
- - - DVD ANIMATION - - -

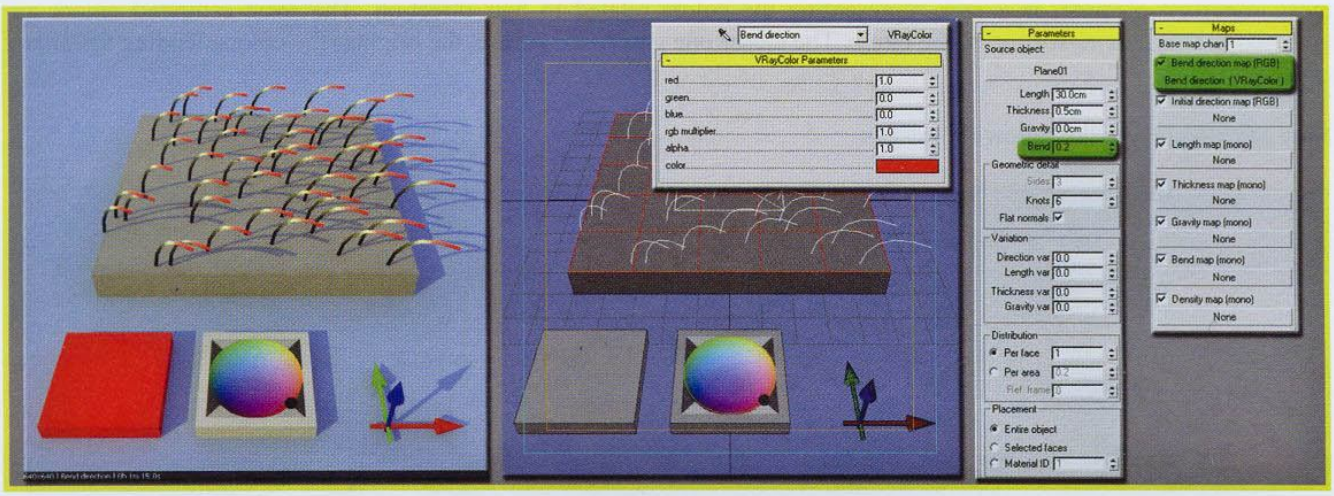
In the previous example filaments are limp. Therefore, it is enough to understand which color allows them to raise. Looking at the texture, it can be noticed that its centre tends to a color very similar to the light-blue: RGB=[128,128,255].

■ **Figure 7.59**
Variation of the vertical inclination by means of the RGB colors.



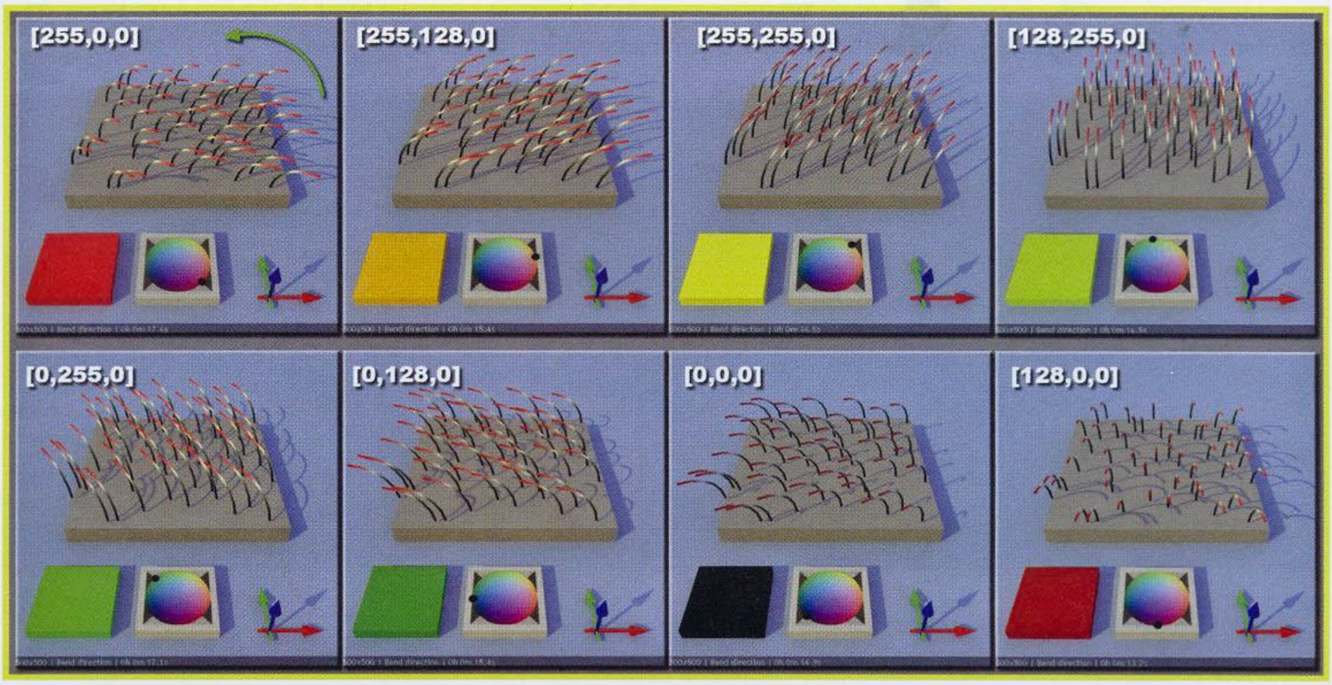
For medium inclination it is enough to move from the outermost colors of the sphere to the innermost ones.

<u>**Bend direction map (RGB)**</u> - as it was for the definition of the ***Initial direction***, for the setting of the bending direction as well it is possible to use an RGB map. In this case the ***Bend*** parameter is longer set to off, but is set to 0.2. In the previous examples it was possible to define the direction of the bending just with the ***Direction var*** parameter. But the direction was randomly managed with it. Instead, by using a map the user has the possibility of precisely defining bending direction.



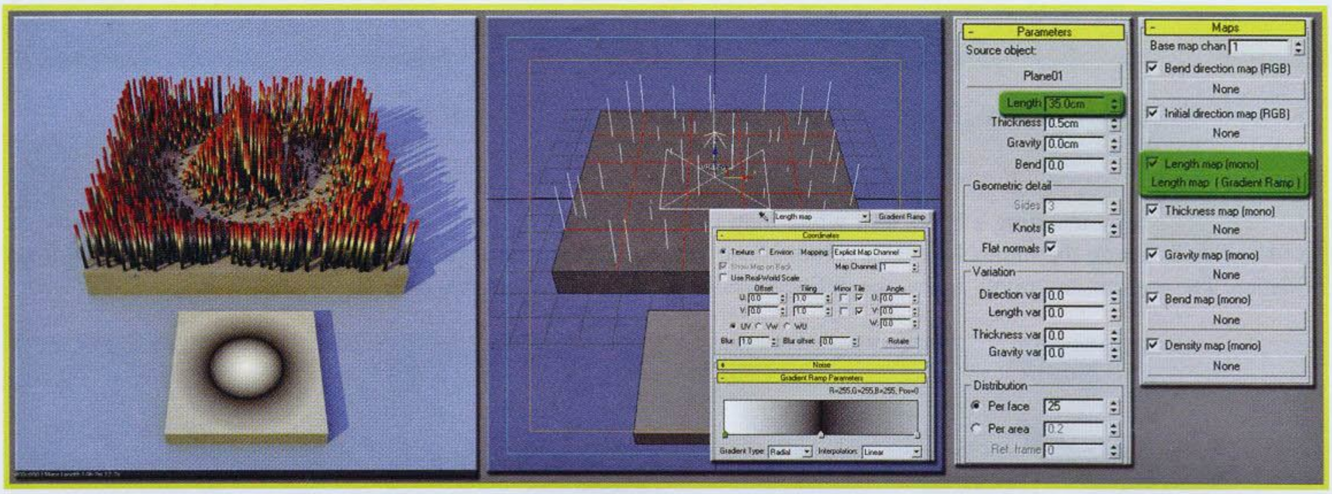
■ **Figure 7.60**
Use of the ***VRayColor*** map, already used as ***Initial direction map***, for the definition of bending direction.

In the example, the direction of the filament is null, appearing perfectly vertical as shown both from the ***Gravity*** = 0 or from the absence of maps in the channel ***Initial direction map***. What varies is the curvature direction, activated by the parameter ***Bend*** = 0.2. Just like the ***Initial direction map***, for the ***Bend direction map*** also, the same observations are valid. To understand the direction of the curvature one just has to look at the colored circular map. Red means the bending is to the right, downwards, while with a bright cyan color RGB = [128, 128, 255] there is no curvature, and filaments are completely straight. The more the color of the ***Bend direction map*** gets closer to the edges of the circle, the closer the curvature gets to the maximum value set in the ***Bend*** parameter.



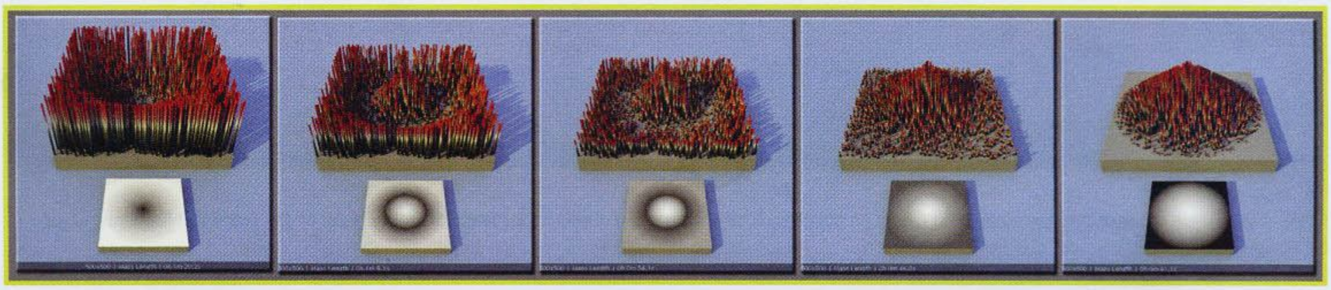
■ Figure 7.61
Graphic representation of the colors corresponding to different curvatures. - - - DVD ANIMATION - - -

Length map - it is possible to define the length of the filaments by inserting a black and white map in this slot. White color means maximum length, while black color means no length. The length is set in the ***Length*** parameter. A circular type Gradient texture is the map used for the definition of the length. As can be seen from the preview in the viewport, the central filaments are the longest ones, while the ring ones are the shortest ones. It is possible to generate animated filaments by animating the gradient parameters.



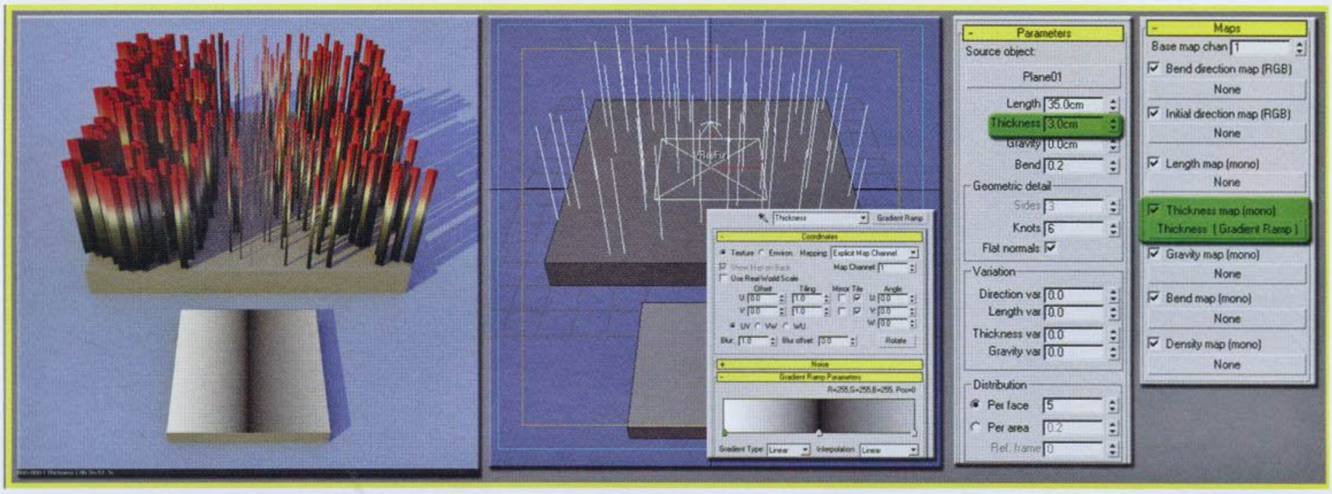
■ Figure 7.62
An interesting composition has been obtained by means of the procedural texture.

Now some frames carried out by animating the gradient map shall be considered.



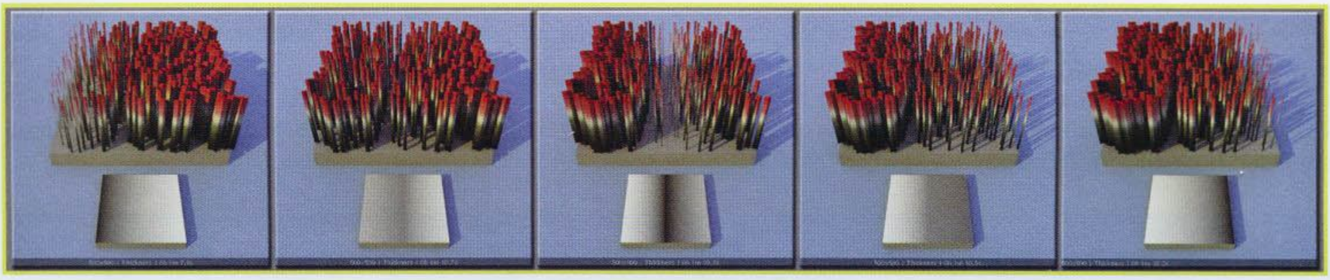
■ Figure 7.63
Frames of an animation with animated filaments. - - - DVD ANIMATION - - -

Thickness map - by inserting a black and white map in this slot it is possible to modify the thickness of the filaments by means of the colors of a texture. White means maximum thickness, and black means zero thickness. The maximum thickness is the one defined by the ***Thickness*** parameter which is 3 cm in this case.



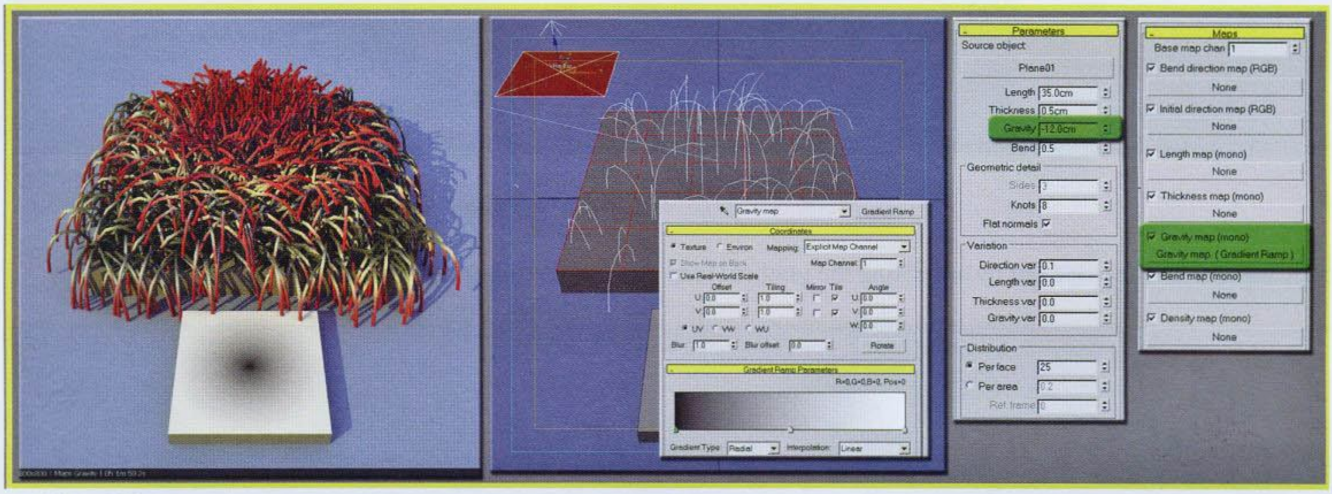
■ **Figure 7.64**
Settings of the Gradient Ramp map of linear type for the definition of the thickness of filaments.

It is possible to obtain interesting effects by animating the map.



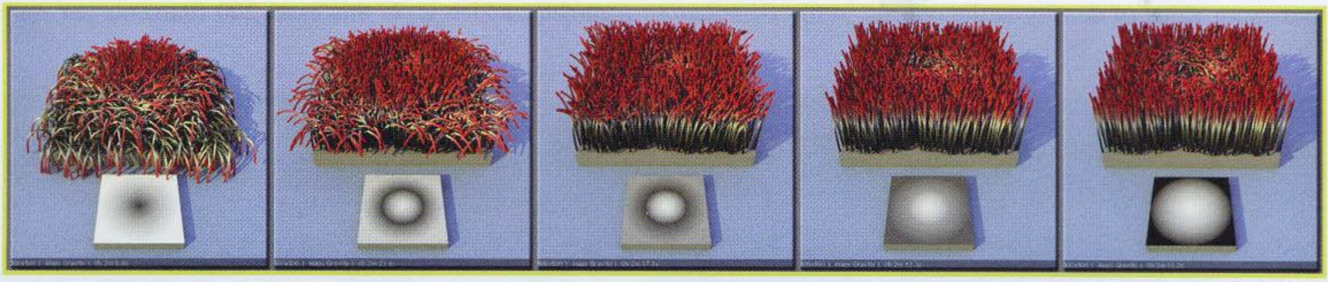
■ **Figure 7.65**
The gradient map applied to the ***Thickness*** channel for the definition of filament thickness.
- - - DVD ANIMATION - - -

Gravity map - by inserting a black and white map into this slot it is possible to modify the influence of gravity on the filaments by using the monochromatic colors of the texture. White means maximum influence, and black means no influence. The maximum gravity is the one defined within ***Gravity***, which, in this case, is -12 cm.



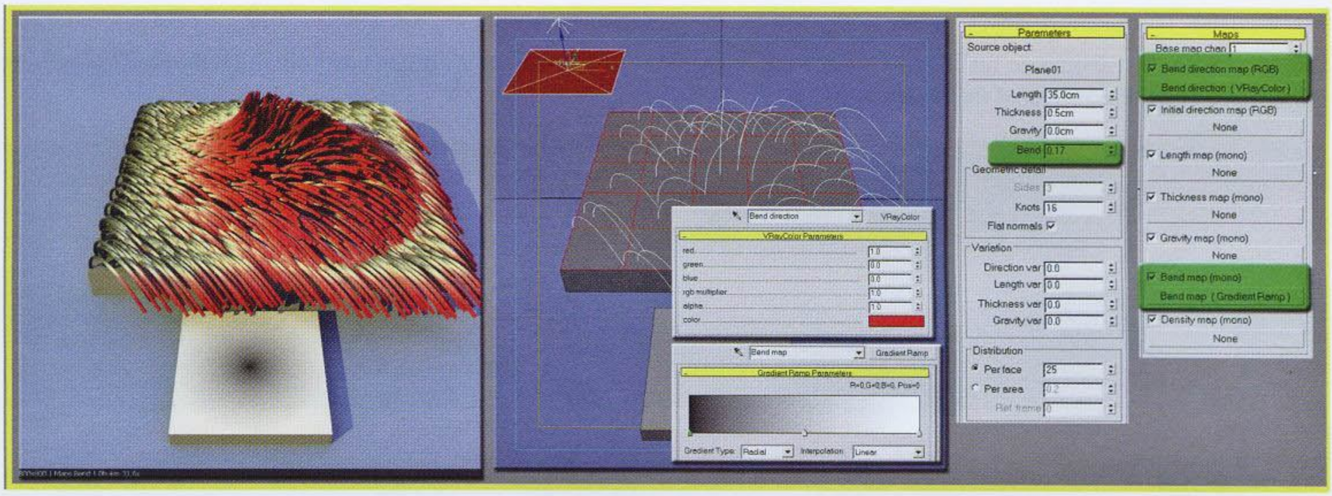
■ **Figure 7.66**
With a radial type gradient map it is possible to create an object such as a flower. The maximum influence of gravity is set on the outermost filaments, where the map is whiter.

The animation of the Gradient map has lead to an interesting animation.



■ **Figure 7.67**
Animation of the gradient texture with consequent variation in the influence of gravity on the filaments.
--- DVD ANIMATION ---

Bend map - by inserting a black and white map in this slot it is possible to modify the bending intensity on the filaments by using the monochromatic colors of the texture. White means maximum bending effect, whereas black means no bending effect. The maximum value of bending is the one set in the ***Bend*** parameter. Two maps are to be used here referred to the previous example. The red ***VRayColor*** map is to be used for the definition of the ***Bend direction***. This way all the filaments are aligned and bend down towards the right, instead of what we saw with the ***Gravity*** map, where filaments had random directions. The second map, the Gradient Ramp, allows one to define the amount of bending of the filaments.

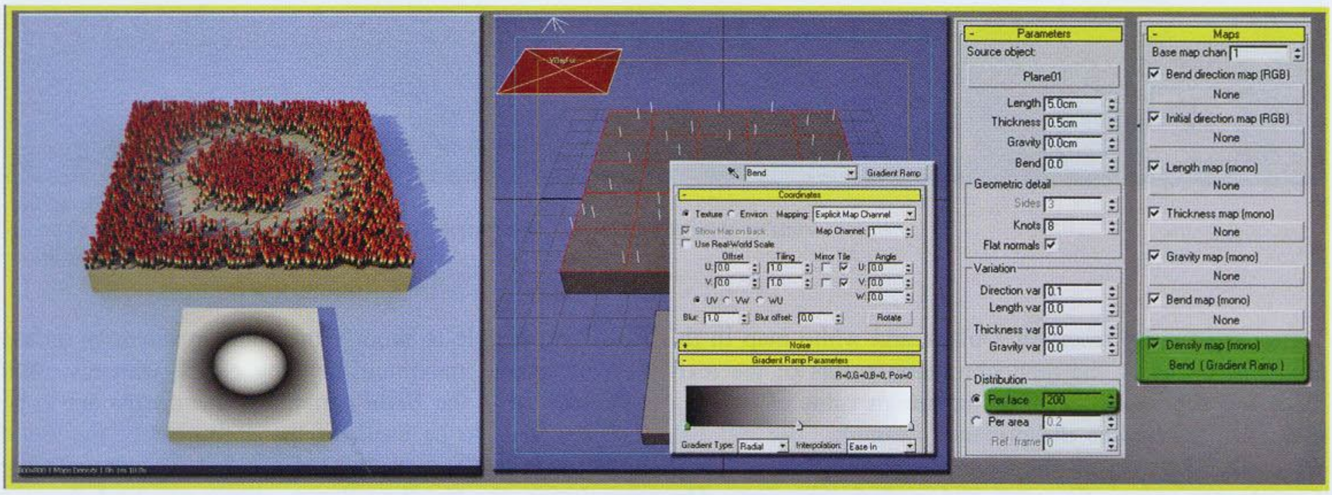


■ **Figure 7.68**
The outermost filaments have a greater degree of bending, the innermost ones tend to be vertical. Furthermore, all the filaments face the same direction.

By animating the map a very interesting "wave" effect is obtained.

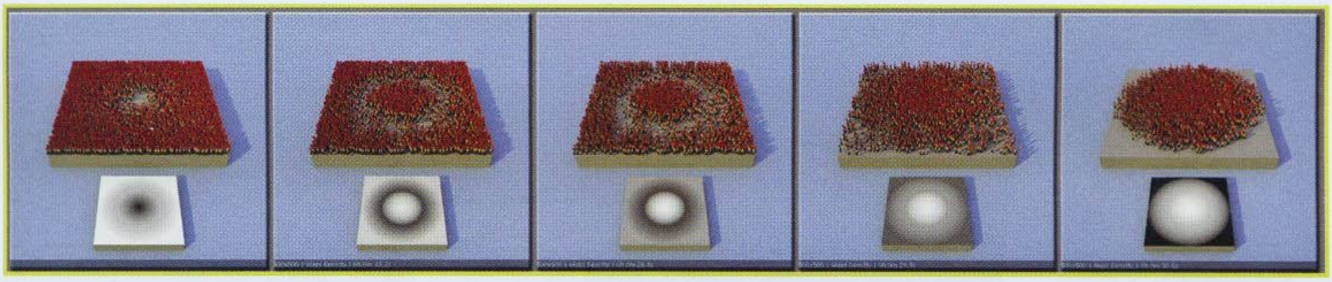
■ **Figure 7.69**
Gradient texture animation with consequent variation in the influence of the gravity on the filaments.
--- DVD ANIMATION ---

Density map - by inserting a black and white map in this slot it is possible to modify the density of the filaments by using the monochromatic colors of the texture. White means maximum density, whereas black means no density. The maximum value of density is the one set in the ***Distribution*** parameter which is, in this case, 200 filaments per each face.



■ **Figure 7.70**
With the ***Density map*** it is possible to set both the arrangement and the amount of filaments precisely.

Lastly an animated sequence fulfilled by animating a gradient map is observed. It has been obtained by modifying both the density and the position of the little filaments.

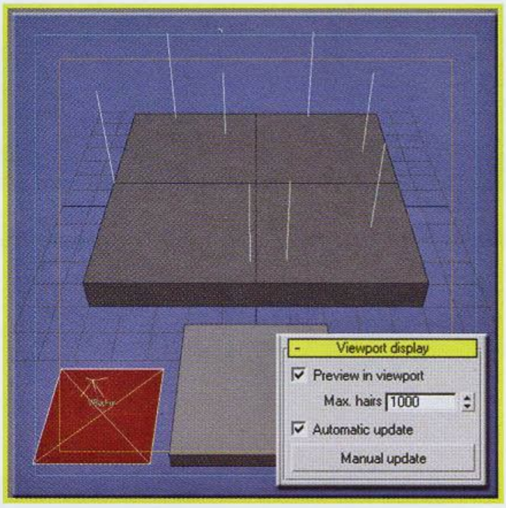


■ **Figure 7.71**
Filament density animation.
- - - DVD ANIMATION - - -

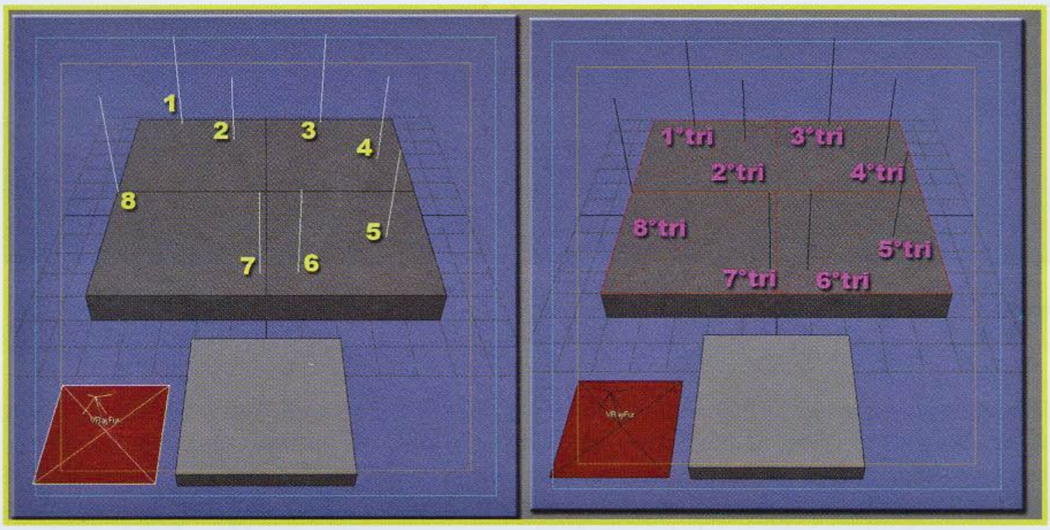
Viewport display

VRay allows one to observe a preview of the ***VRayFur*** by means of a preview composed by simple spline directly in the viewport.

■ **Figure 7.72**
Preview in the viewport of the filaments and rollout for the definition of the control parameters of the preview.

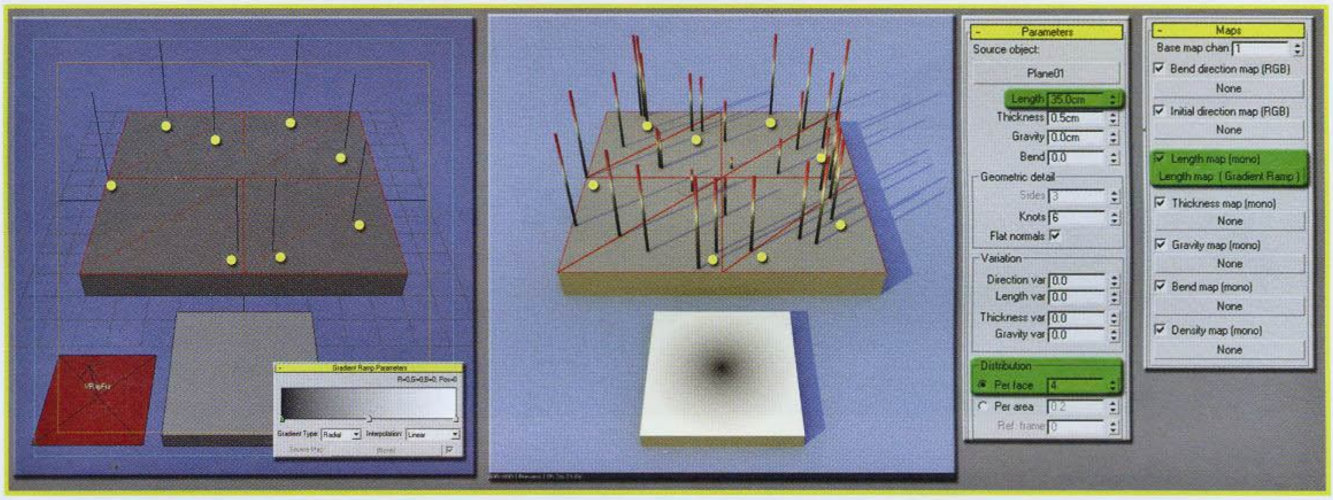


To know how many filaments appear in the preview it is enough to know that *VRay* assigns one filament for each triangular face. In the previous case, since four squares, correspondent to 8 triangles, are present, the total number of filaments in the preview is 8, regardless of the maximum number of filaments visible in the viewport, a value which is managed by the ***Max hairs*** parameter.



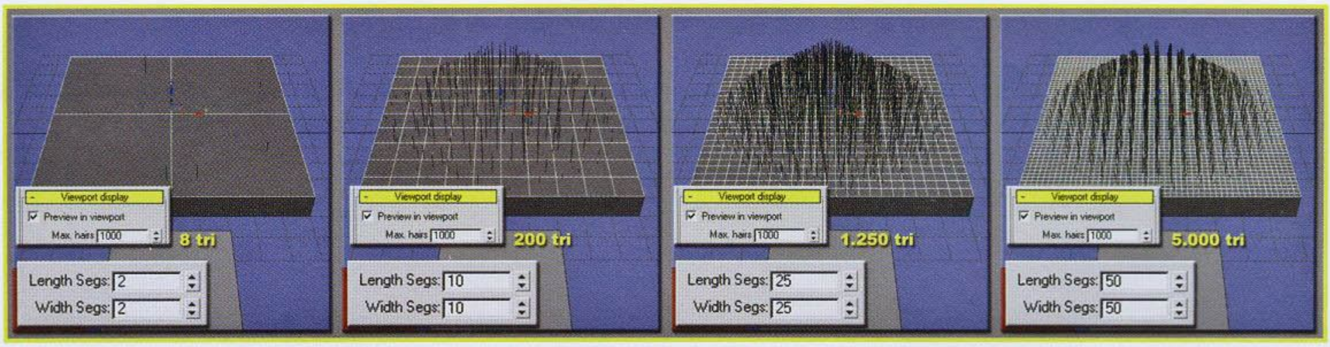
■ **Figure 7.73**
Scheme of the number of filaments in the viewport.

By rendering the image and using four filaments ***Per face***, the result is 32 filaments totally. Notice that the position of the filaments in preview actually corresponds to the rendered ones.



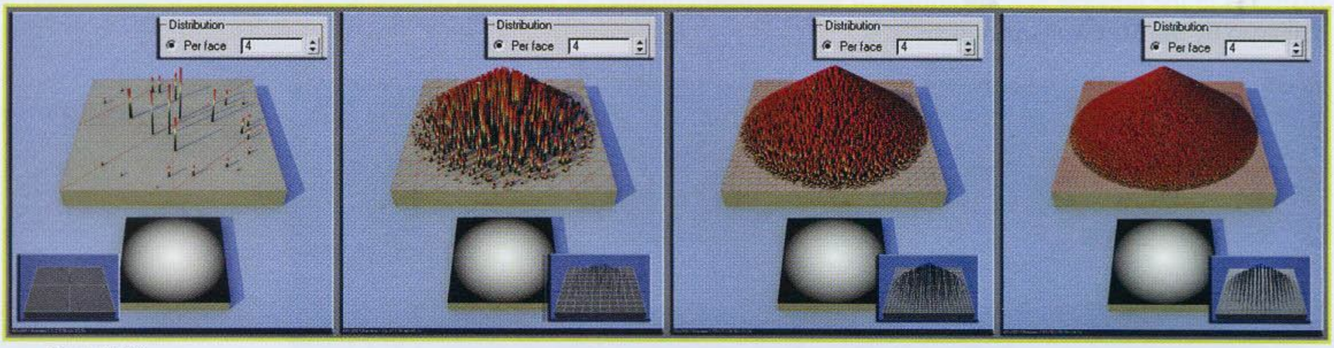
■ **Figure 7.74**
Correspondence between the preview filaments and the final render.

Filaments do not have the same length since a procedural map has been used for defining this property. Unfortunately it is not possible to observe directly in the viewport the different lengths, due to the small number of filaments. For this purpose it is necessary to increase the plane subdivision. In such a way the filaments in the viewport increase, allowing one to observe the different lengths in real-time.



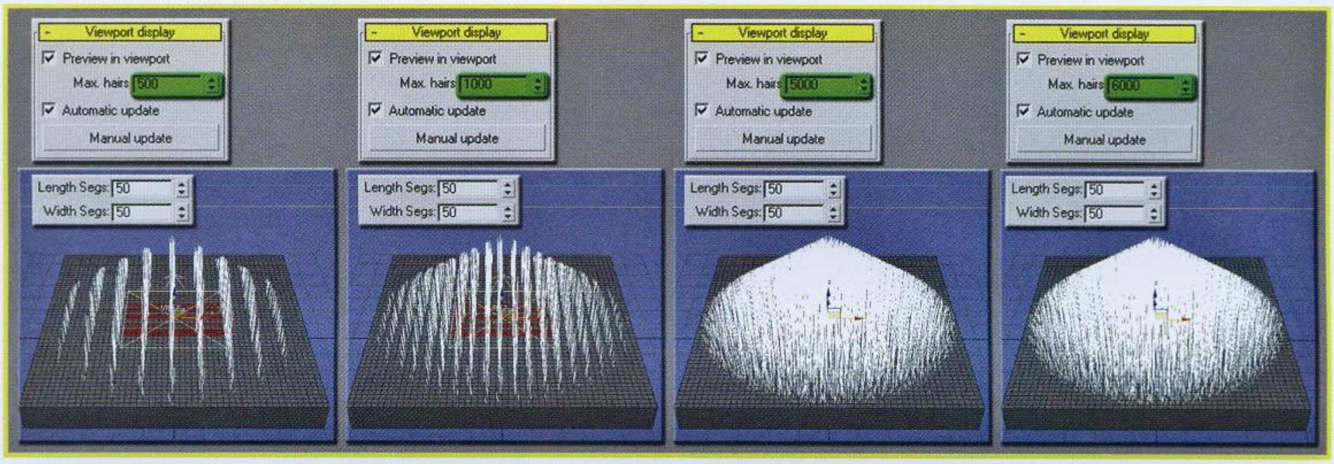
■ **Figure 7.75**
Increase in the number of filaments in preview after the increase of the emitter's polygons.

Obviously, by increasing the number of segments of the emitter's mesh, the ***Distribution Per face*** being equal, the number of rendered filaments increases as well.



■ **Figure 7.76**
Variation of the density of filaments due to the increase of the emitter's subdivision.

Increasing the number of filaments, the amount of filaments in the viewport increases as well. But since the maximum number of visible filaments has been set to 1,000, no other hairs are going to be added in preview, once this threshold is passed, as happens with the plane subdivided 25x25 or 50x50. The difference between the planes of 25x25 and 50x50 is the different arrangement of the filaments, but their number on the viewport remains 1,000 in any case. In these cases, in order to obtain a better output in the working window, it is necessary to modify the ***Max. hairs*** parameter respectively to 1,250 and 5,000.



■ **Figure 7.77**
Variation in filament density due to the increase of the ***Max hairs*** parameter.

In order to obtain the best preview possible with a geometry composed of 5,000 triangles, 5,000 filaments are necessary. Higher values do not produce any change. Values lower than 5,000 for the ***Max hairs*** reduce the amount of filaments in preview.

Preview in viewport - it activates or deactivates the preview in the viewport.

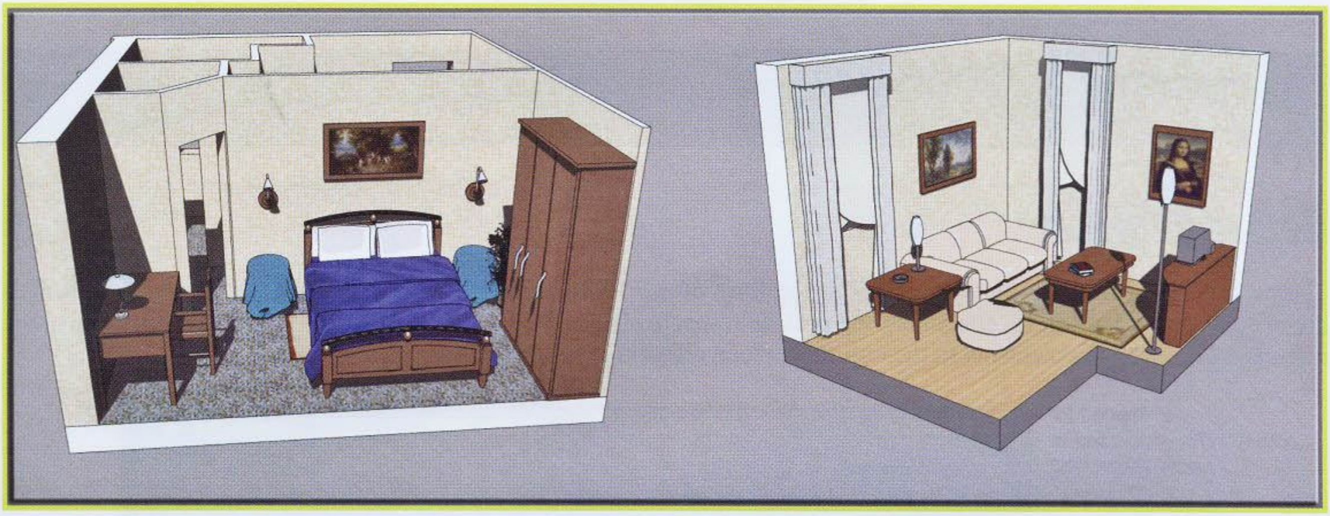
Max hairs - it defines the maximum number of filaments visible in the viewport.

Automatic update - by activating this option the variation of the ***Max hairs*** parameter automatically produces a check in the viewport. By deactivating it is necessary to click on the ***Manual update*** to update the preview.

VRayToon

INTRODUCTION

VRay is software specifically conceived for photorealistic images. But other types of renders, called **NPR** (**N**on **P**hotorealistic **R**endering) exist, aimed at representing 3D models in a "cartoon" style.

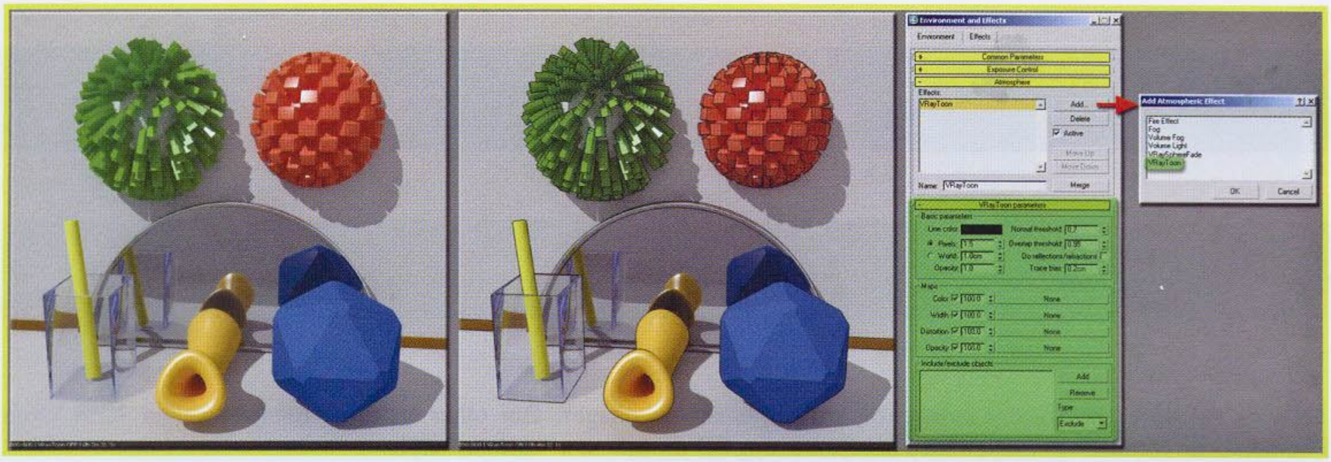


■ **Figure 7.78**
NPR render.

These effects can be generated essentially with two techniques. The first one involves the employment of shaders, materials purposefully studied for this kind of output. The second one, instead, uses post-production operations, that is it works when the render is finished. *VRay* uses the second method. It can be said this is not the best method for this kind of renders, and ***VRayToon*** is not in fact a properly complete NPR system, but, in any case, it allows one to realize renders with hidden lines with, in addition, effects such as the management of line thickness, of the color, the transparency and the distortion. By using various techniques it is possible to obtain "tract" effects with this system. Plug-ins exist to realize NPR renders, as finalToon or Illustrate specifically created for this kind of output. The reasons which have led to this kind of solution are:

- Very simple and user friendly implementation of the toon system within 3ds Max.
- It allows one to render any geometry supported by *VRay*, including ***Displacement*** and ***VRayFur***.
- It works with any kind of camera, such as the spherical type, ***Fish eye***, cylindric, etc...
- It works with any camera effect, like ***DOF*** and ***Motion Blur***.
- It works in presence of reflecting or transparent surfaces.
- It allows one to generate very accurate contours between intersecting surfaces.

Using the atmospheric ***VRayToon*** effect is very easy. One just has to build the scene as one usually would; afterwards ***VRayToon*** must be activated. Automatically the edges of the meshes, called "outlines", are highlighted with a black line. To activate ***VRayToon*** it is necessary to enter the Environment panel, open the Atmosphere rollout and click on the button Add. At this point the ***VRayToon*** effect is selected. A new rollout appears with all the ***VRayToon*** parameters. Moreover we will explain how to modify this effect in order to create all kinds of lines.
In the next image some objects, among which ***VRayFur***, ***Displacement***, transparent, reflecting, organic and sharp meshes are used. This scene allows one to observe the behaviour of the ***VRayToon*** effect in relation to very different geometries.



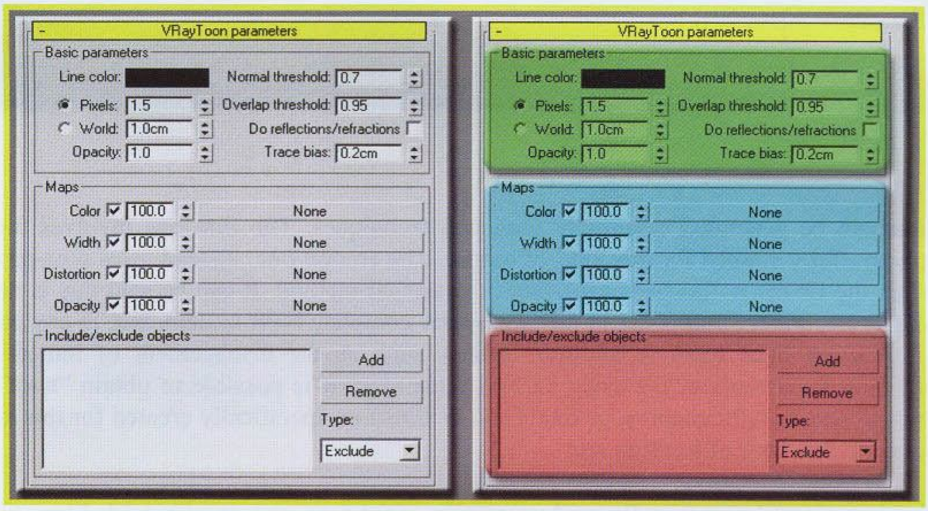
■ **Figure 7.79**
Application of the ***VRayToon*** effect to a photorealistic kind of scene. Notice the thin contour lines.

PARAMETERS

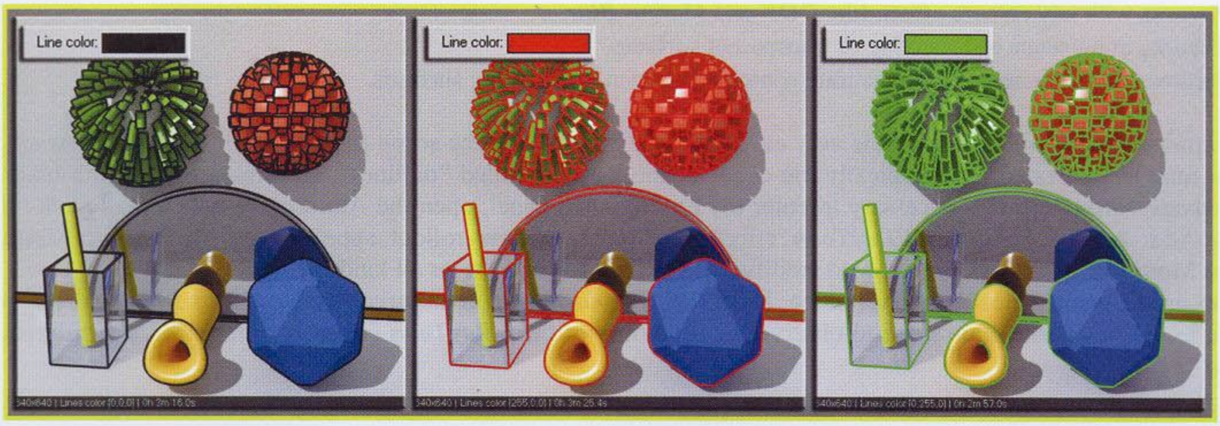
Basic parameters

The parameters controlling the appearance of the lines of the ***VRayToon*** effect are contained in just one rollout. It is divided in three macro-areas, containing the options respectively for the parametric control of the lines, the control via maps and the exclusion/inclusion of the objects the toon effect is to be applied to.

■ **Figure 7.80**
The ***VRayToon*** rollout and its control parameters.



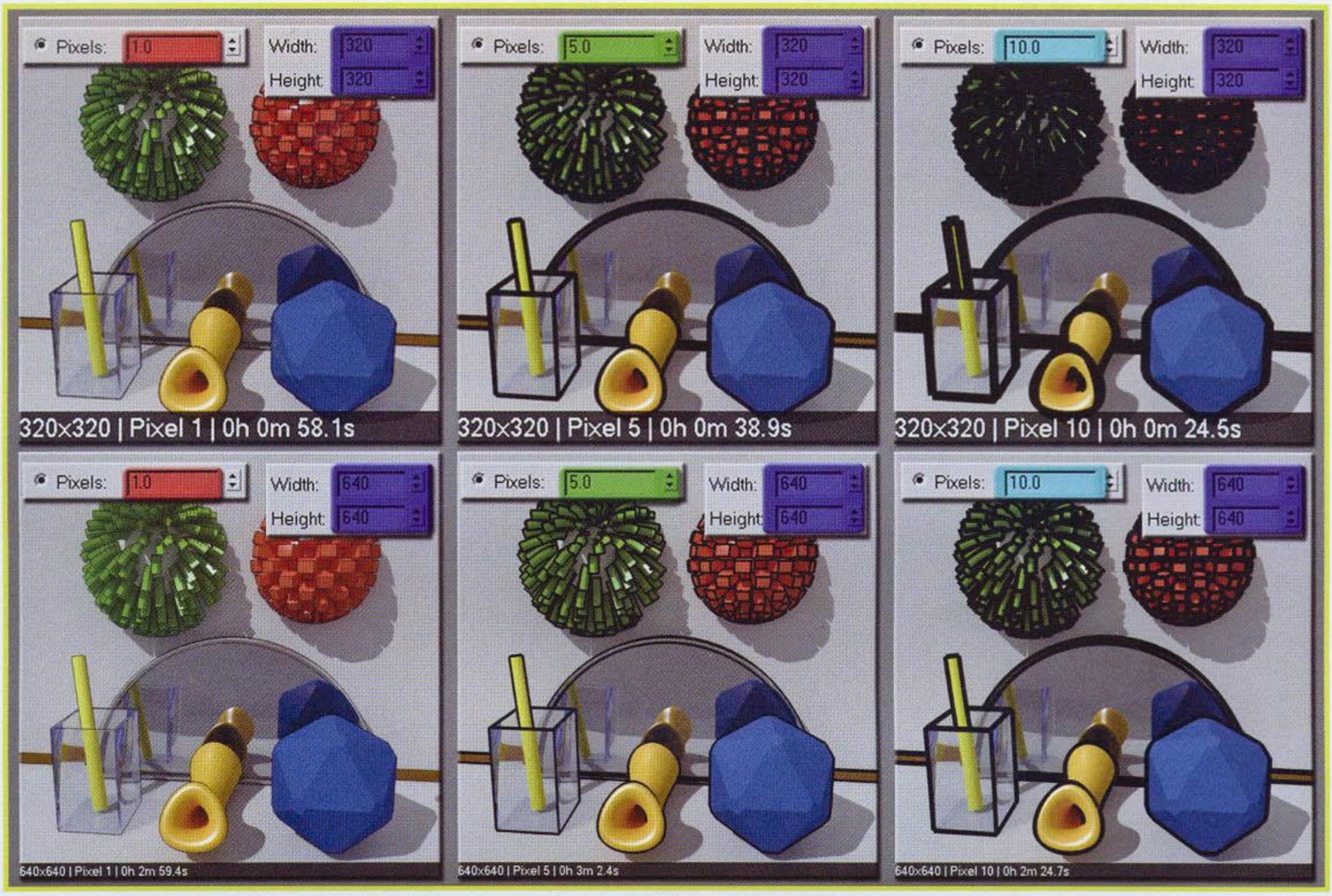
Line color - by modifying the color in this panel it is possible to change the color of the outlines.



■ **Figure 7.81**
Variations of the color of the outlines. The thickness of the lines has been increased for the test, specifically from 1.5 pixels up to 5 pixels.

Pixels - two measuring units exist for the definition of the thickness of the contour lines: the ***Pixels*** unit, and ***World*** unit. The ***Pixels*** unit calculates the thickness of the contour lines using the pixels measurement unit. The employment of ***Pixels*** has two main consequences:

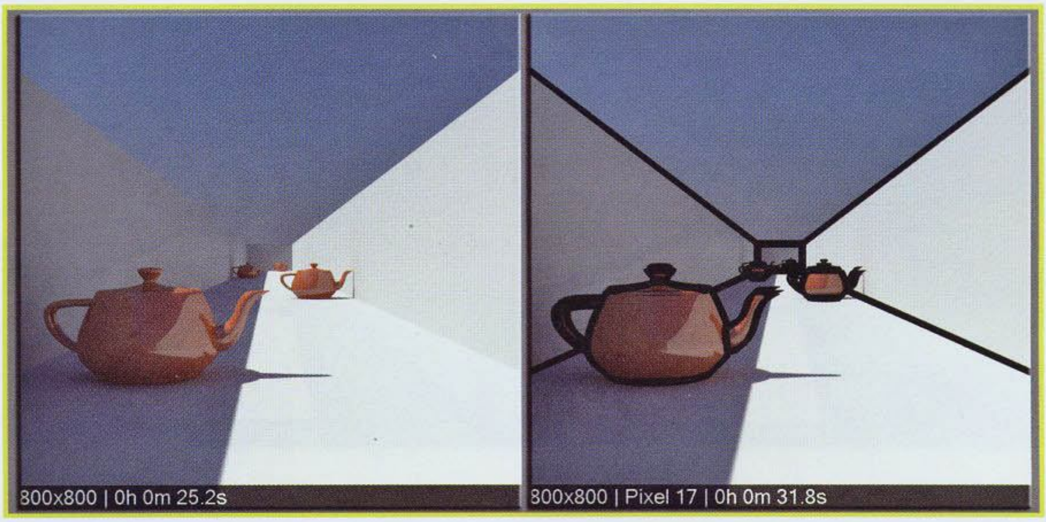
1. Varying the output resolution, the thickness of the lines measured in pixels changes. Keeping the thickness unvaried and increasing the render's resolution only, the line, compared to the objects on the scene, is smaller, while its size in pixels remains unvaried.



■ **Figure 7.82**
Analysis of the Pixel parameter with two different resolutions: 320x320 pixels and 640x640 pixels.

In this test the same render, with the same settings for the ***VRayToon***, has produced two different effects. In the example with thickness 10 pixels applied to the ***VRayFur***, the filaments appear completely black, due to the high thickness, at the resolution of 320x320 pixels, instead at 640x640 pixels the green color of the filaments is visible.

2. The thickness of the line is always constant, regardless of the distance from the observer's point of view.

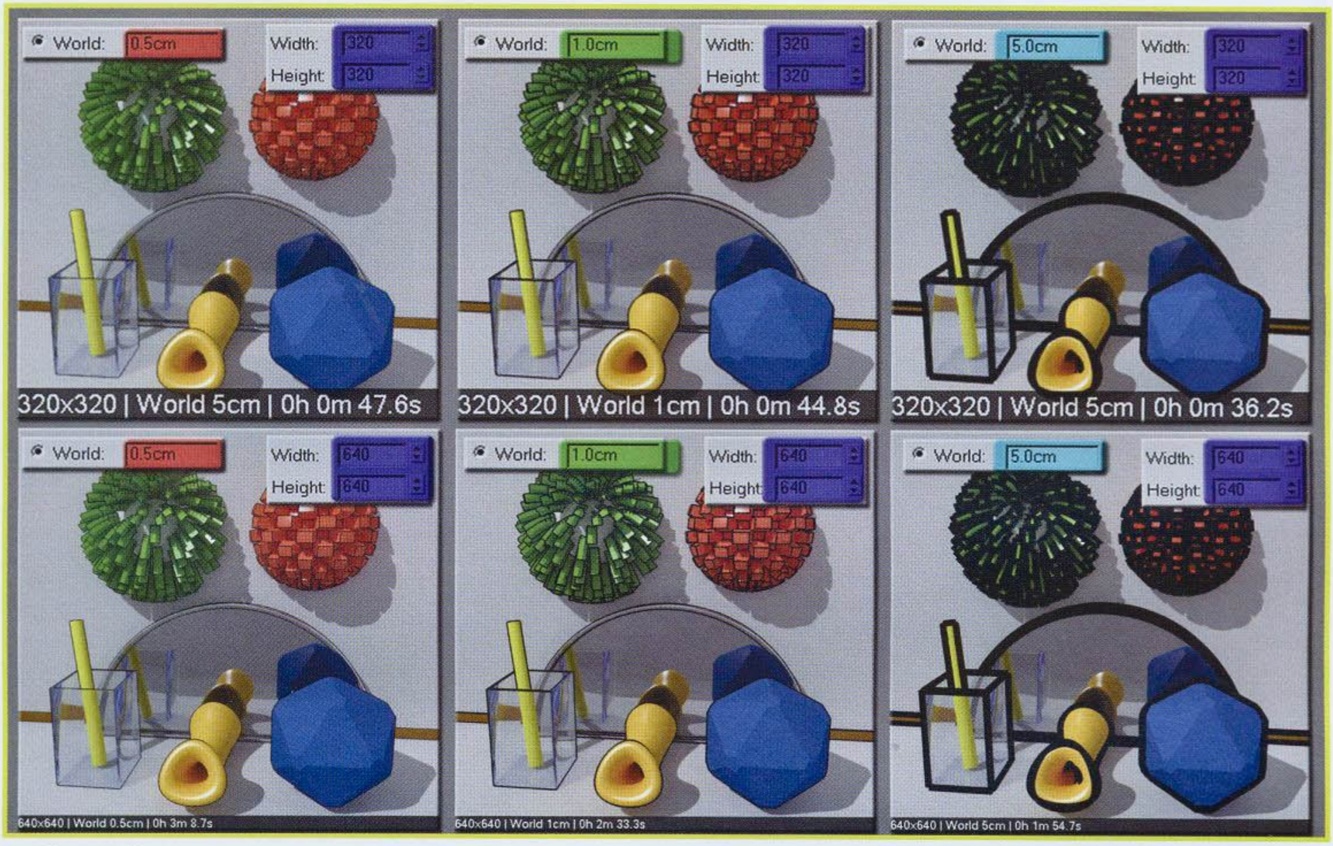


■ **Figure 7.83**
A corridor is an example of geometry having parts which are nearer and further away from the observer's point of view. The lines have constant thickness.

The lines of the corridor, both near and far from the observer, like the wall at the end of it, always have the same thickness. In this case the lines are 17 pixels thick.

World - this parameter allows one to use as unit of measurement the one used by the 3ds Max file, in this case the cm. the ***World*** unit has some opposite features compared to the previous one.

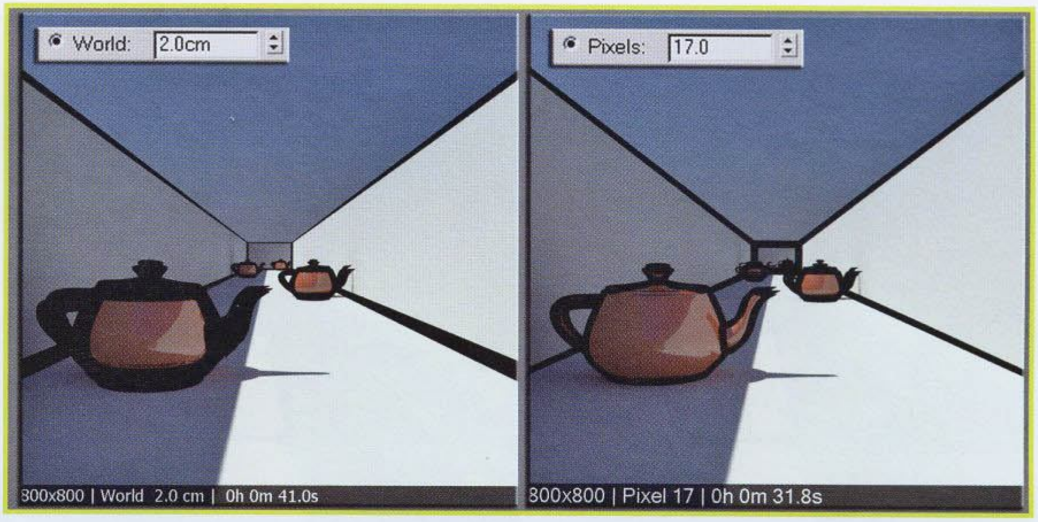
1. When varying the rendering resolution the thickness of the lines does not change. Keeping the ***World*** thickness unvaried and increasing the render resolution the thickness of the lines, relative to the objects on the scene, is always the same. Visually, lines with the same degree of thickness are obtained, independently on the output resolution.



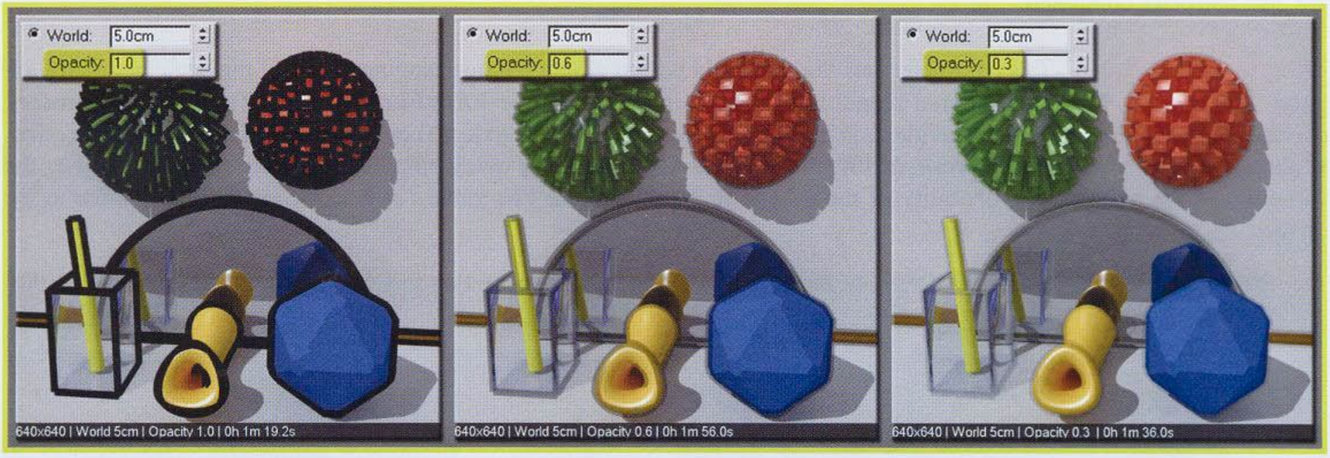
■ **Figure 7.84**
Images are the same, both at 320x320 pixels resolution and at 640x640 pixels resolution.

2. The thickness of the line is variable in accordance with the distance from the point of view. Since the unit of measurement is represented in real sizes, as one gets further away from the point of view the objects or, in this case, the lines, must become smaller in order to maintain the same size. It is the same as what happens with the samples of the ***Light Cache***.

■ **Figure 7.85**
Differences between the employment of the ***World*** and ***Pixels*** units. Both on the outliner of the tea-pots both on the corridor, getting further away from the observer's point of view, with the ***Pixels*** unit the thickness of the lines remains unvaried. This does not happen with ***World***.

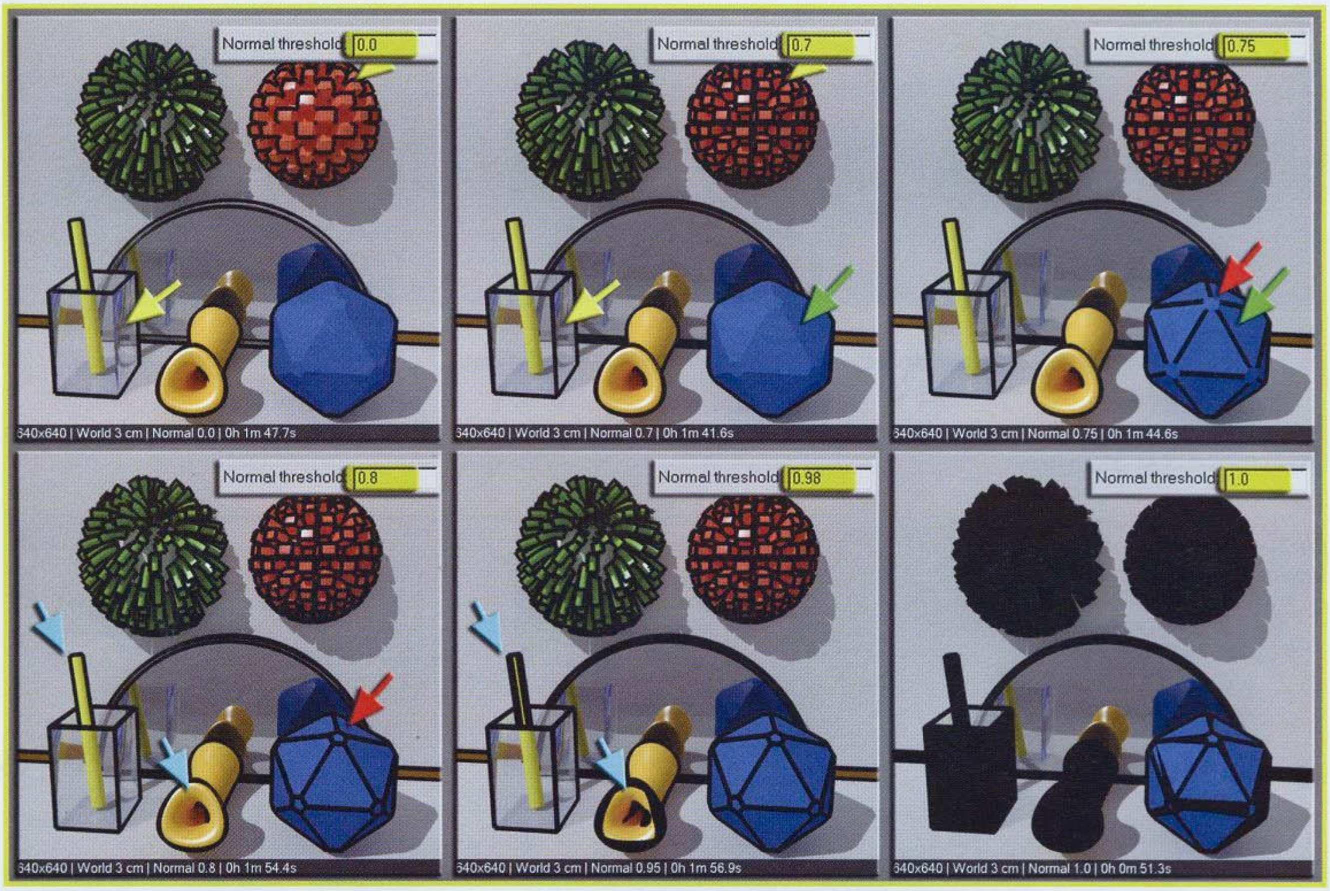


Opacity - this parameter controls the opacity of the outlines. Values lower than 1.0 make the lines more and more transparent.



■ **Figure 7.86**
The ***Opacity*** parameter and the transparency of the contour lines generated by the ***VRayToon*** effect.

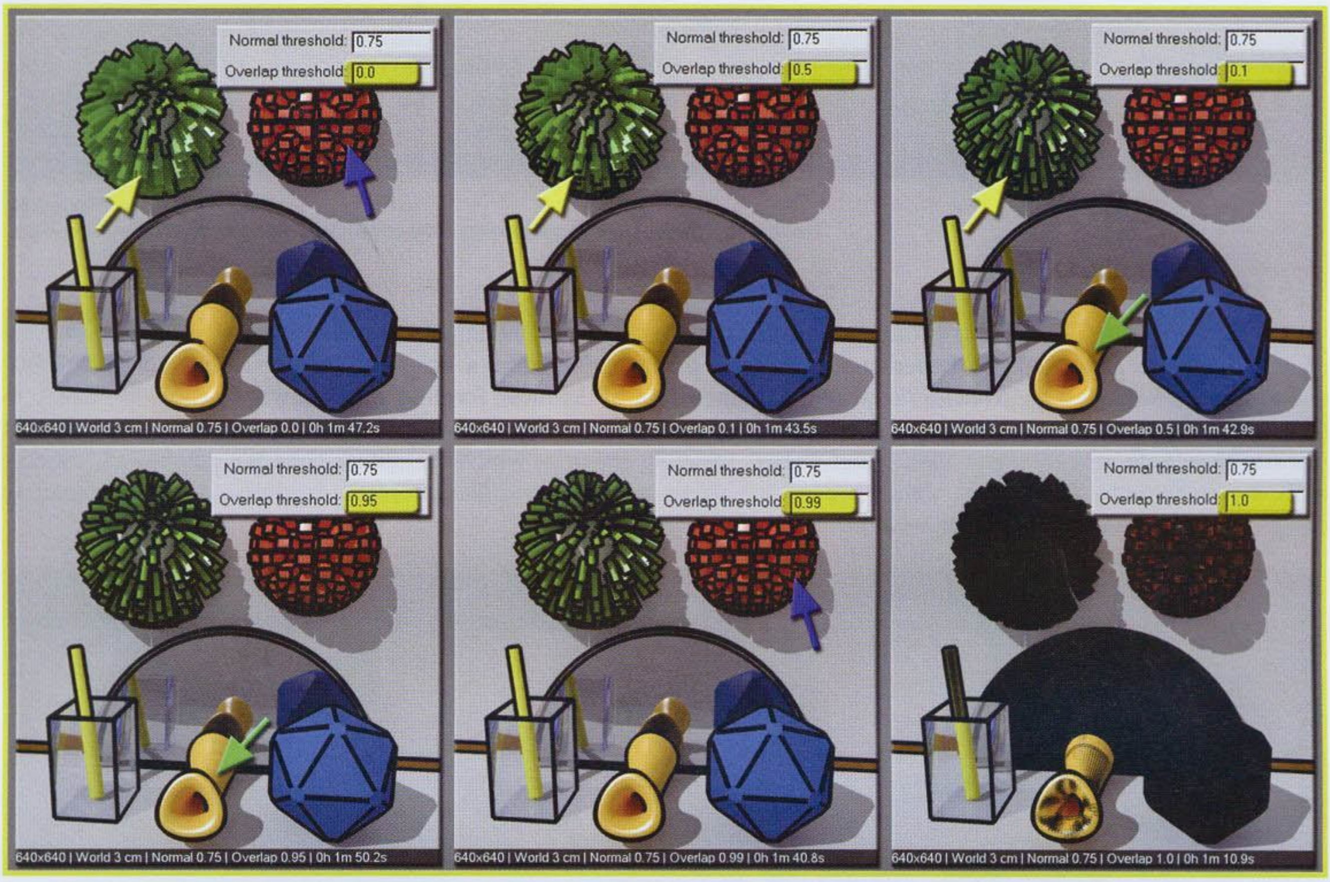
Normal threshold - with this parameter it is possible to determine the angular threshold below which the internal contour lines are generated. To better understand this parameter imagine a box. This geometry has sides with 90° angles. With a value 0.0 for ***Normal threshold***, only the edges of the mesh forming angles lower or equal to 90° are to be rendered by ***VRayToon***. When we say angles lower than 90°, this means sharper edges. By increasing the value of ***Normal threshold*** also the faces with wider angles, meaning the ones which are obtuse (like the patches of a soccer ball), are rendered by ***VRayToon***. Values around 1.0 are to be avoided: in fact, problems might occur, like completely colored faces or edges rendered incorrectly.



■ **Figure 7.87**
Definition of the contours between faces with different angles.

In the previous example the colored arrays mark the variations due to the increase of the ***Normal threshold*** parameter. Starting from the glass with smooth angles close to 90°, passing to the dodecagon, finishing with the organic geometry, one can see the various phases of the tracing of the internal edges. By increasing ***Normal threshold*** the flattest angles are also correctly computed.

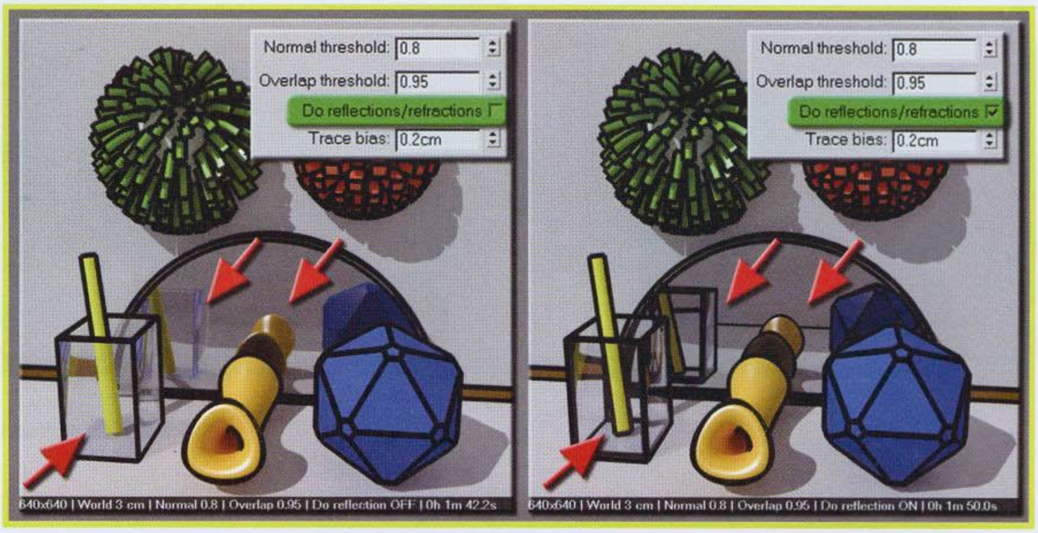
Overlap threshold - this parameter sets the threshold within which the overlapping edges of the same geometry are going to be traced. In the considered scene this situation is visible on the ***VRayFur*** geometry. The filaments, in fact, overlap. Another situation, not so evident, is in the central tube, where the external entrance overlaps the central section. With low values of ***Overlap Threshold,VRayToon*** does not render the overlapping edge. Vice versa, the higher the value is and the higher the probability of the outline to be calculated. The value 1.0 is to be avoided, as shown in the image.



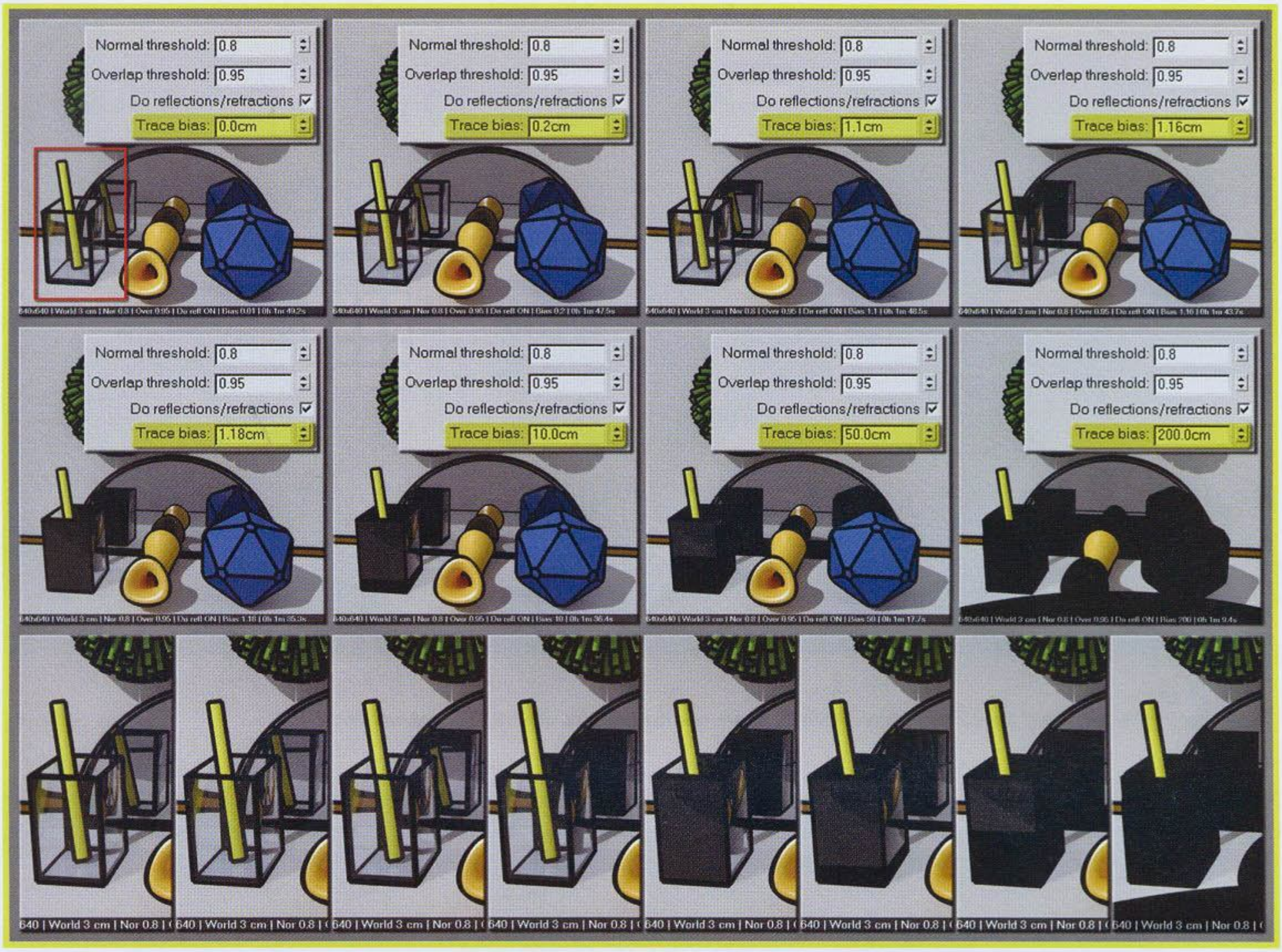
■ **Figure 7.88**
The management of the outlines of overlapping meshes is controlled by the ***Outline threshold*** parameter.

Do reflection / refraction - the activation of this parameter generates outlines both on reflecting objects both inside transparent geometries.

■ **Figure 7.89**
Notice the edge lines in the mirror, in the transparent glass and in the straw. With the ***Do reflection / refraction*** option active these objects are correctly rendered.



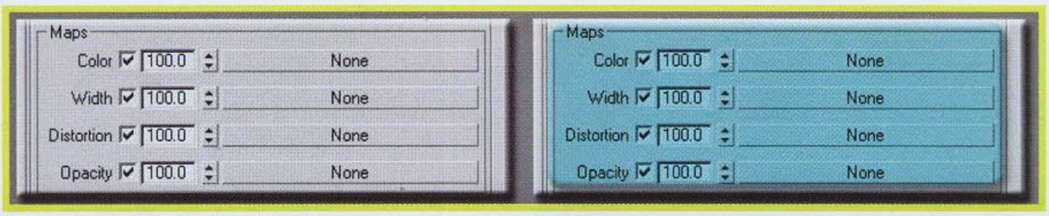
Trace bias - this parameter, affected by the scale of the scene, determines the bias when the edge lines are calculated on reflecting or transparent objects. The higher the ***Trace bias*** is set, the more artefacts could be found. It is advisable to try to keep this value as low as possible.



■ **Figure 7.90**
The ***Trace Bias*** parameter.

Maps

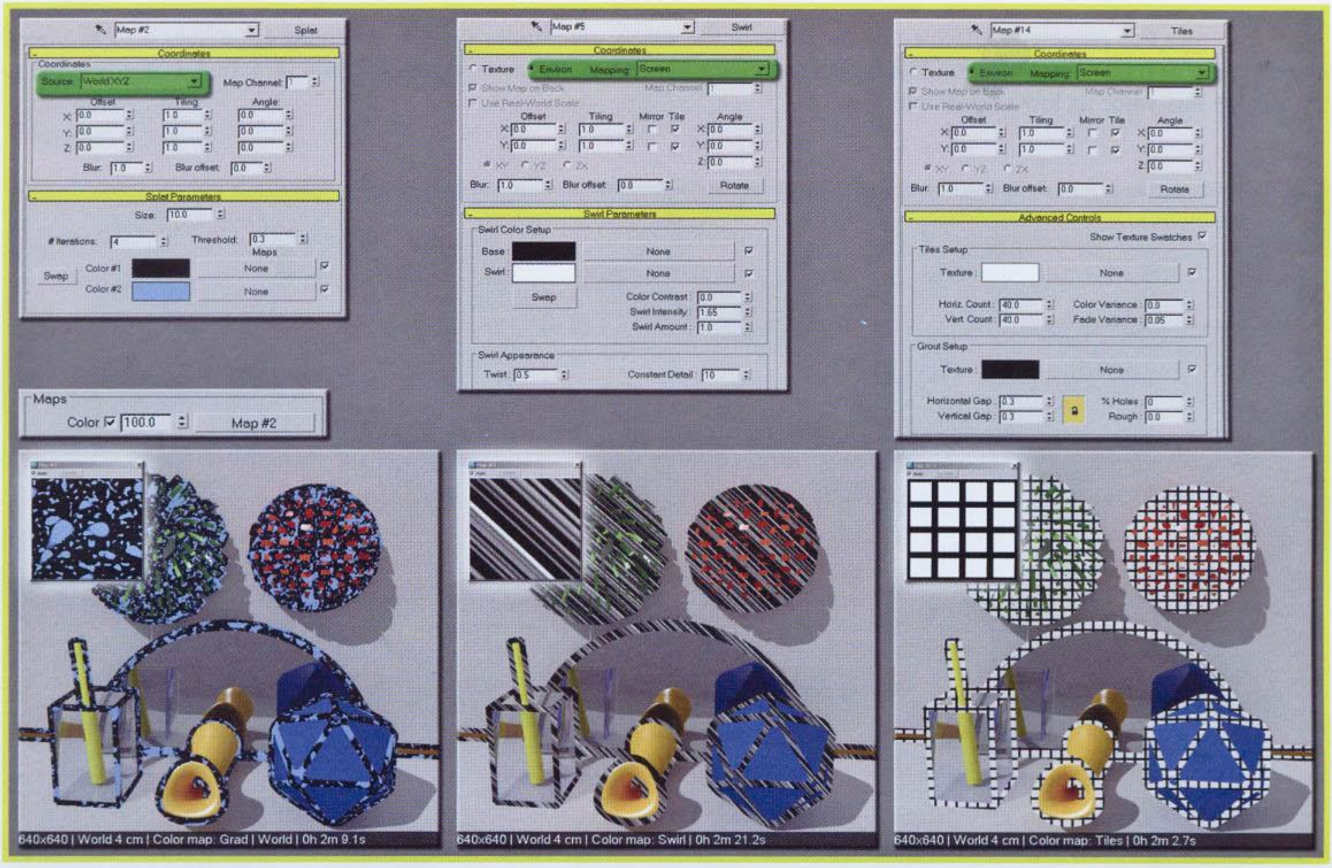
Until now the properties of the edge lines produced by the ***VRayToon*** effect have been managed and modified by means of numerical values except for color. ***VRay*** allows one to manage these options through maps and textures as well, in the same way ***VRayFur*** was managed.



■ **Figure 7.91**
The four maps for ***VRayToon*** management.

The four maps regard the management of color, distortion and opacity.

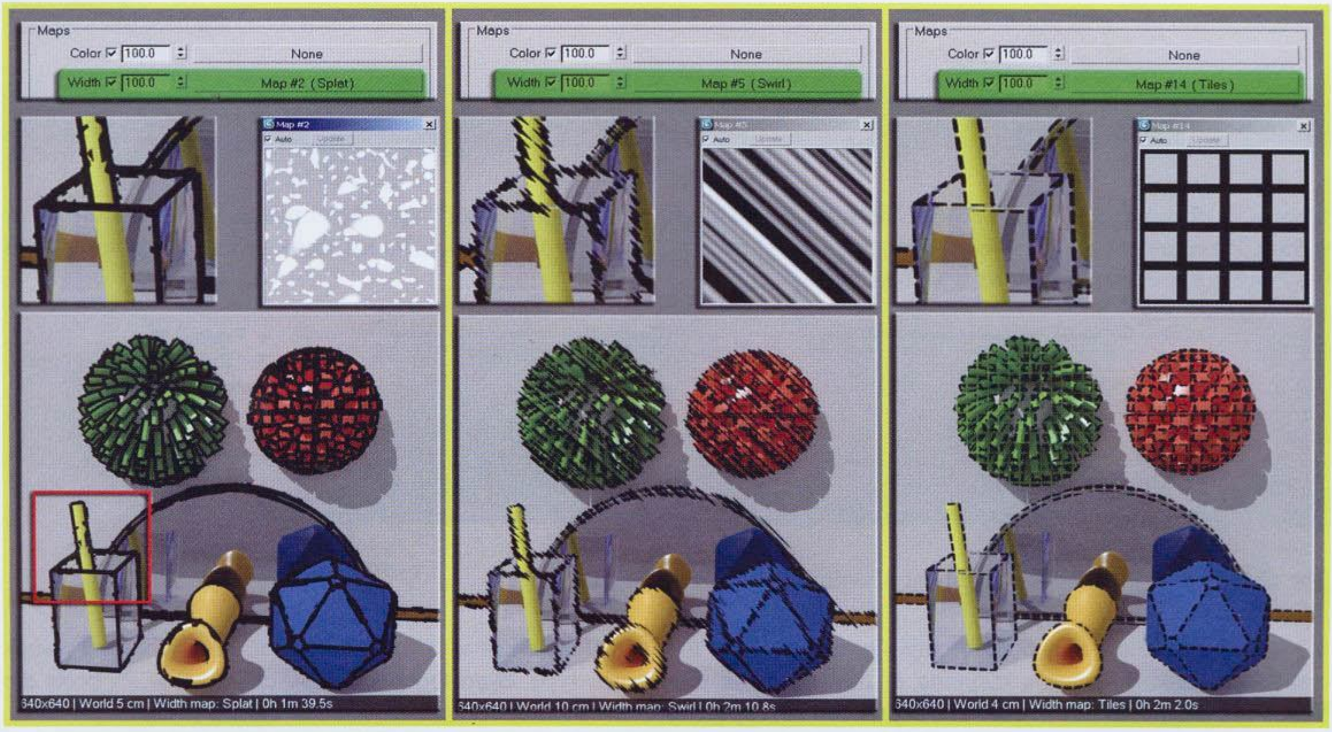
Color - by inserting a map inside this slot it is possible to manage the outlines color through a texture. The best results can be obtained by using a mapping of the texture of the ***Screen*** type, even though, in any case, ***VRay*** supports the World XYZ type, useful in case of 3D procedural maps.



■ **Figure 7.92**
Some maps for the definition of the color of the lines. Notice the Screen type mapping used with the 2D procedural maps.

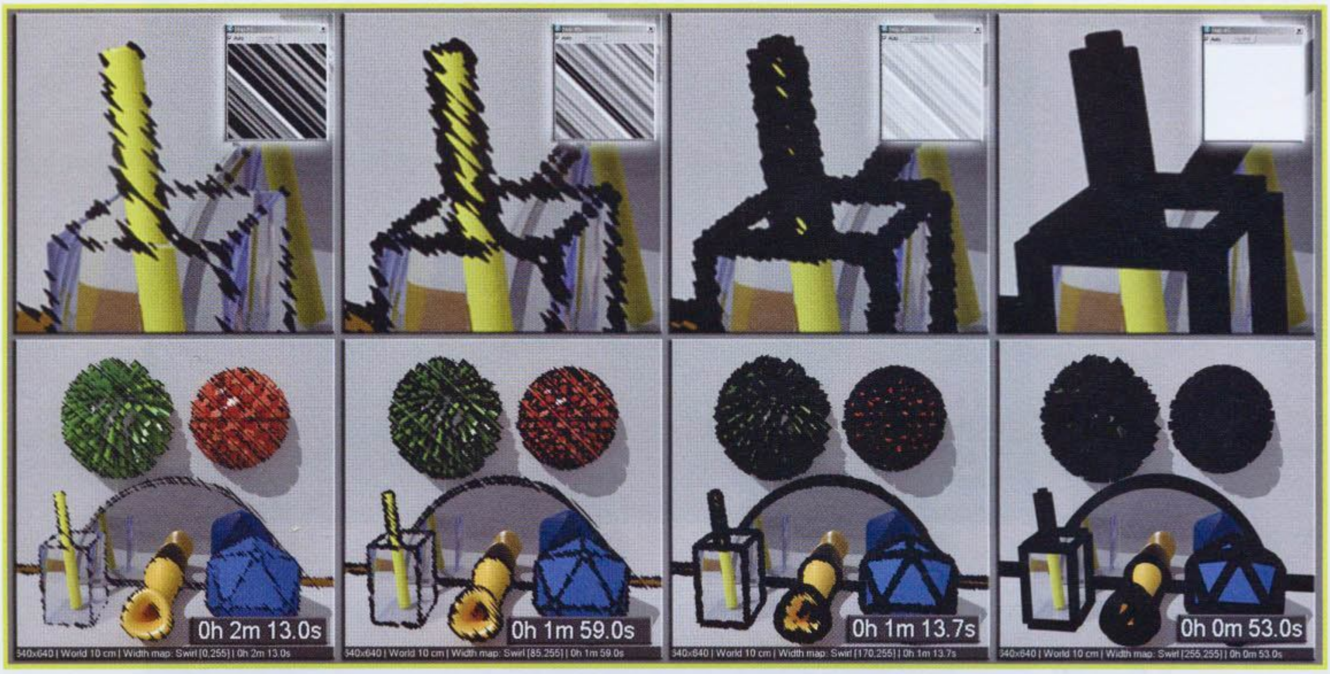
Next to the Color label a checkbox is present. Once deactivated, the color of the lines is managed only by the ***Line color*** color previously analyzed, independently of the presence of a map in the ***Color*** channel. Finally, with the spinner, which controls the percentage of use of the map, it is possible to mix the texture and the ***Line color*** color.

Width - with this map it is possible to manage the width of the lines by using a black and white map.



■ **Figure 7.93**
Different types of outlines obtained by inserting some procedural maps inside the ***Width*** channel.

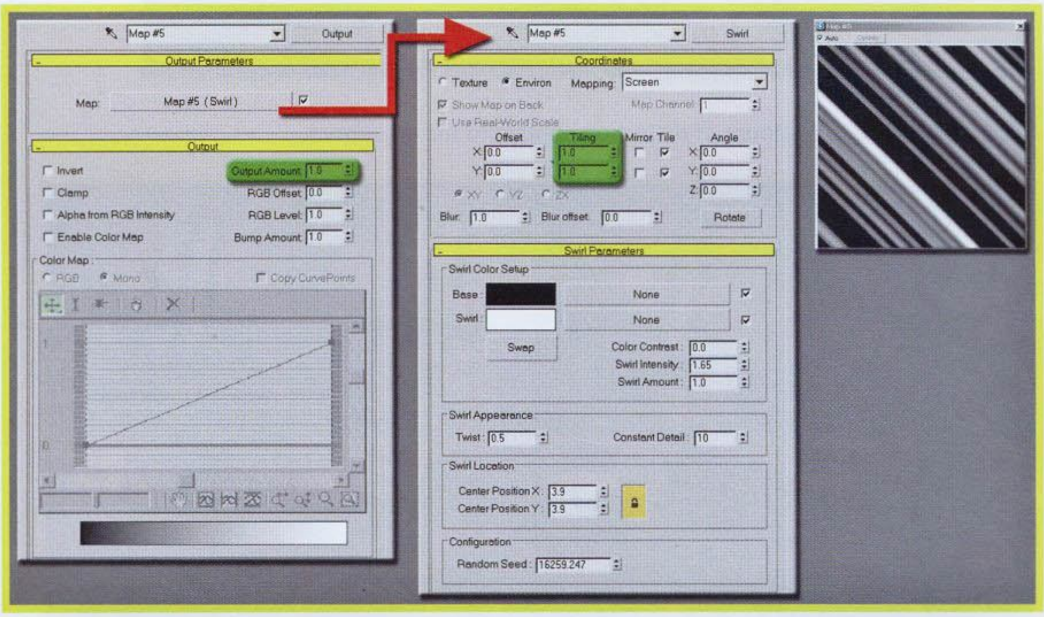
It is possible to obtain interesting types of outlines by inserting a black and white map in the thickness channel. The more the color of the map nears white, the more the outline is opaque. Vice versa, the more the color of the map tends towards black, the more the tract appears transparent.



■ **Figure 7.94**
The same type of the tract can be modified by changing the color from white to black. The higher the contrast between bright and dark areas is, the higher the rendering time is to be.

Like the ***Color*** map, in this case too, the checkbox and the numerical spinner allow one to mix the tract between the edge lines, created with the parameters of the ***Basic parameters*** section, and the outlines modified by the texture inserted in the Width channel.

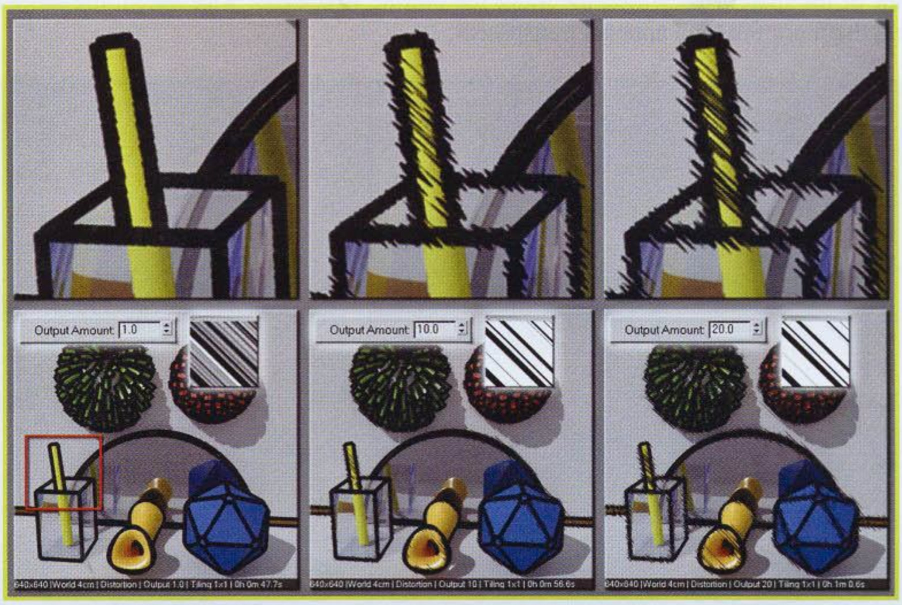
Distortion - with this option it is possible to define the type and the amount of distortion in the edge lines. The procedure is slightly different to the previous maps. In addition to the 2D, 3D or raster procedural maps, it is necessary to use, in fact, the Output Amount parameter.
In the next test the Swirl procedural map is going to be used. This map is preceded by the Output map. It is possible to modify the Output Amount parameter with it. As for the Swirl map, it is interesting to notice the variation in the outlines after the change of the Tiling parameter. In any case, it is possible to modify all the other parameters as well, for instance the color of the map into black and white, therefore affecting the ***VRayToon*** result.



■ **Figure 7.95**
The map used inside the ***Distortion*** channel.

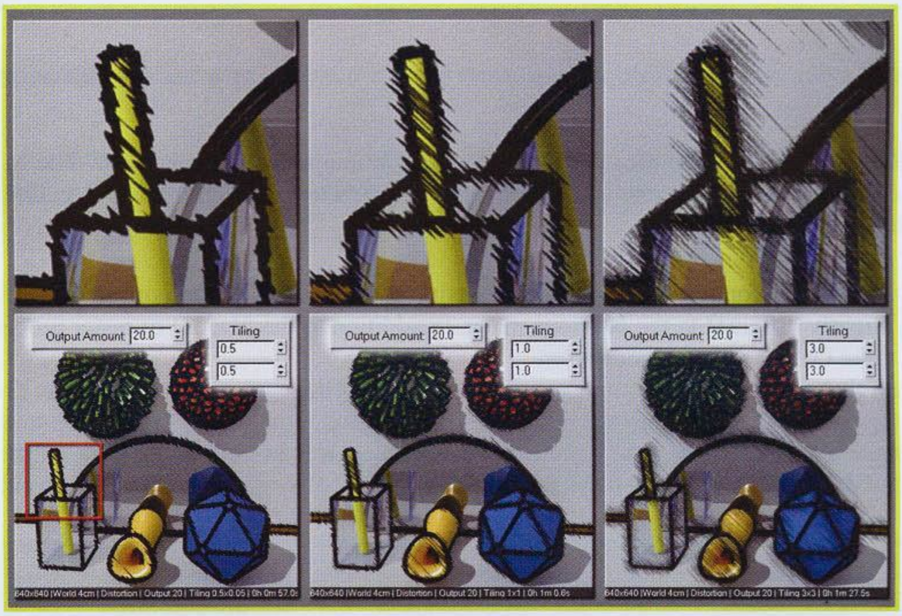
In the next image the influence of the Output Amount parameter on the deformation of the line can be observed.

■ **Figure 7.96**
Different distortion levels obtained by having increased the Output Amount parameter.



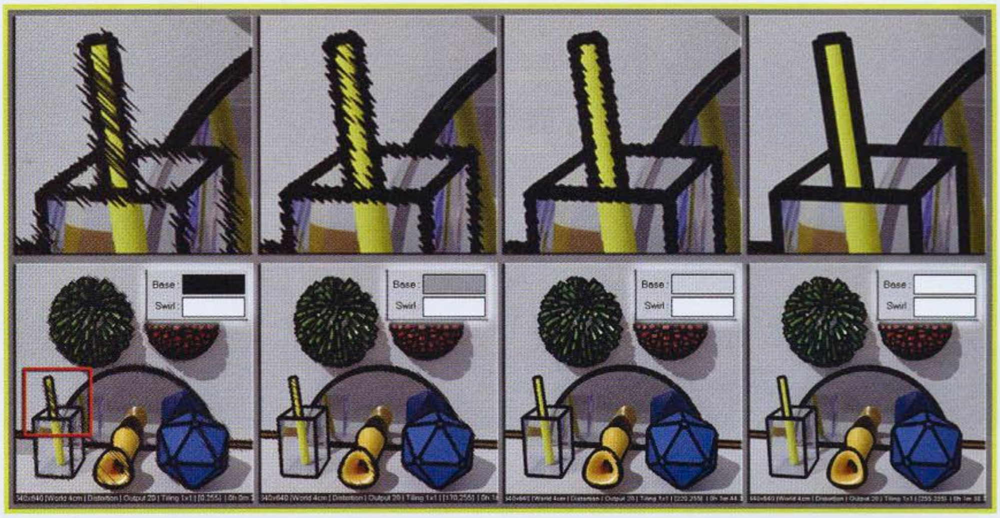
By modifying the Tiling parameter of the procedural map a thickening of the lines is obtained.

■ **Figure 7.97**
Increase in the "vibrations" of the outlines.

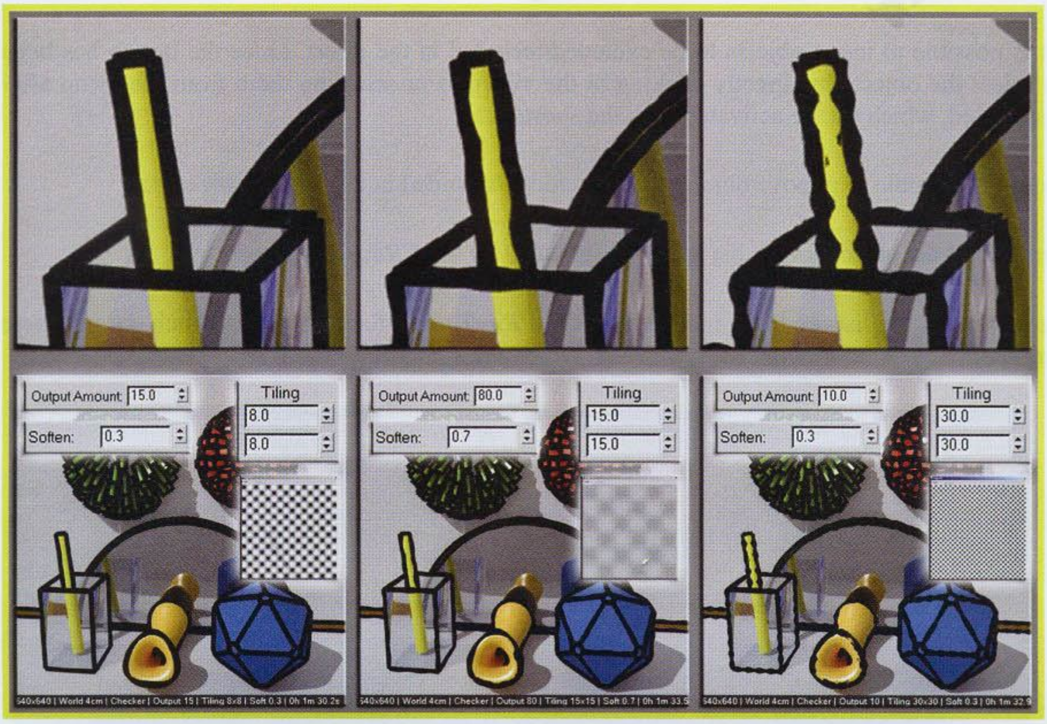


With the color of the map one can work on the amount of distortion applied to the edge lines.

■ **Figure 7.98**
Soft lines obtained by varying the colors of the Swirl map.

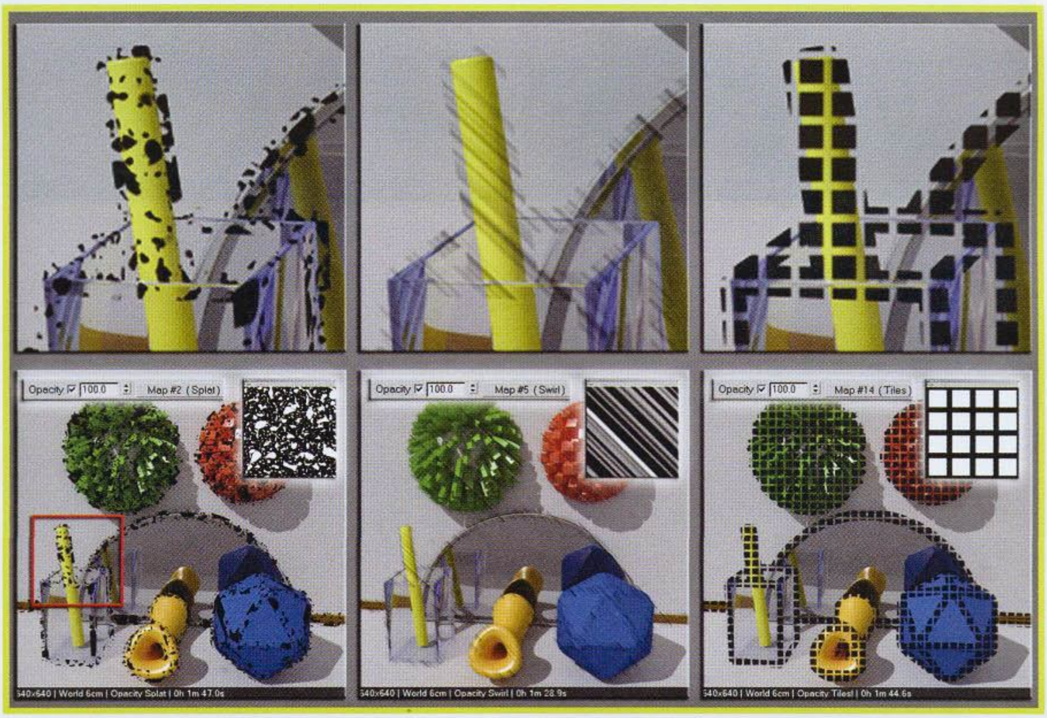


Other examples fulfilled with the Checker map. The softness of the lines can be noticed, typical of a "cartoon" effect.



■ **Figure 7.99**
Creation of wavy edge lines thanks to the employment of a procedural map and its Soften parameter.

Opacity - by inserting a procedural map in this slot it is possible to control the opacity of the lines. White means complete opacity. Black means complete transparency.



■ **Figure 7.100**
Special effects obtained with the opacity of contour lines.

In this case as well, un-checking the option near ***Opacity***, *VRay* uses the option located in the ***Basic parameters*** section for determining the transparency of the lines. Instead, by using the spinner, the effect obtained with the ***Opacity*** map is mixed with the ***Opacity*** parameter located in ***Basic parameters***.

Include/exclude objects - with this table it is possible to include or exclude the geometries chosen by the user from ***VRayToon*** computation.

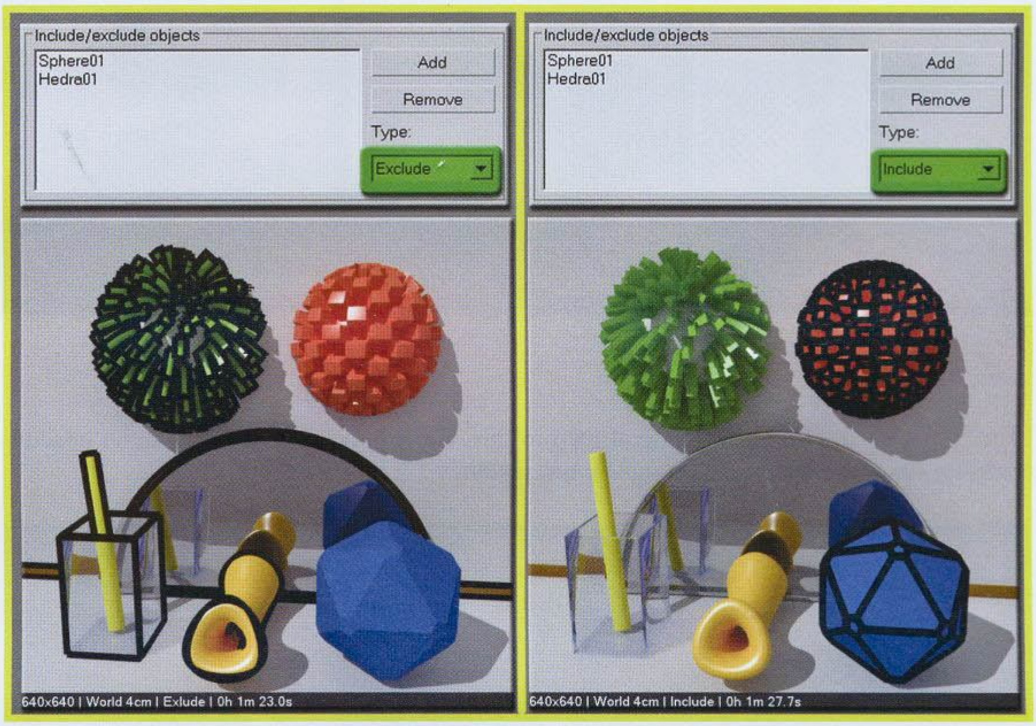
Add - with this button it is possible to insert objects to be excluded/included in the effect. Once the button has been activated, it is possible to select the objects by directly clicking in the viewport or selecting them from the menu after the Pick Object table has appeared, which can be activated with the shortcut H.

Remove - with this button it is possible to remove objects to be excluded/included in the ***VRayToon*** effect.

Exclude - with this command all the objects in the list are to be excluded by the ***VRayToon*** effect.

Include - all the objects present in the list are to be included by the ***VRayToon*** effect. The ones which are not there are not rendered with ***VRayToon***.

■ **Figure 7.101**
By excluding the sphere with the ***Displacement*** and the dodecagon, these geometries are not inserted into the toon effect. Vice versa, if the ***Include*** option is selected, the two geometries in the list are going to be the only two rendered by ***VRayToon***.



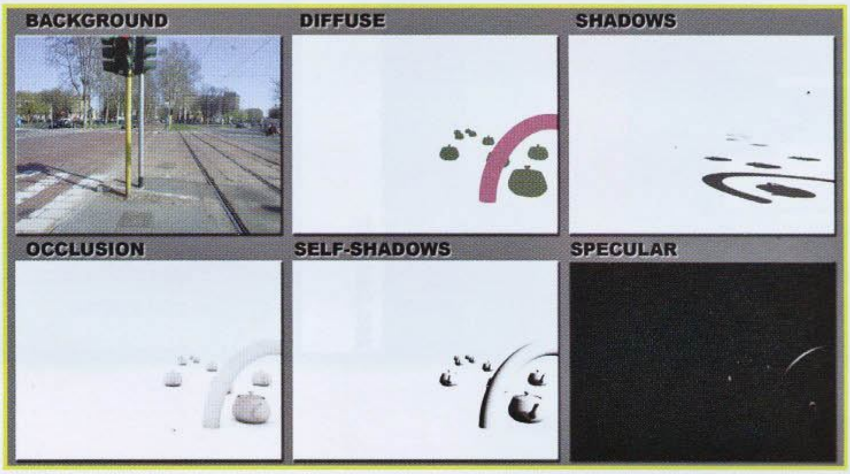
VRaySphereFade

INTRODUCTION

This topic, compositing, represents a unknown subject for those who are approaching the 3D world for the first time. Generically the term video compositing refers to a branch of video editing (2D and 3D video) which has become essential, in the last years, in the professional management of digital videos. In fact, it is a procedure allowing one to modify some visual effects of a video by adding special effects by means of overlapping several video sources. The whole of this is carried out by using layers, additional video channels which can be thought of as transparent glossy sheets overlapped on the original movie. New clips can be put on the layers, creating complex effects easy to manage. Layers can be moved, rotated, modified at will. The advantage is that higher levels are not affected by these changes.

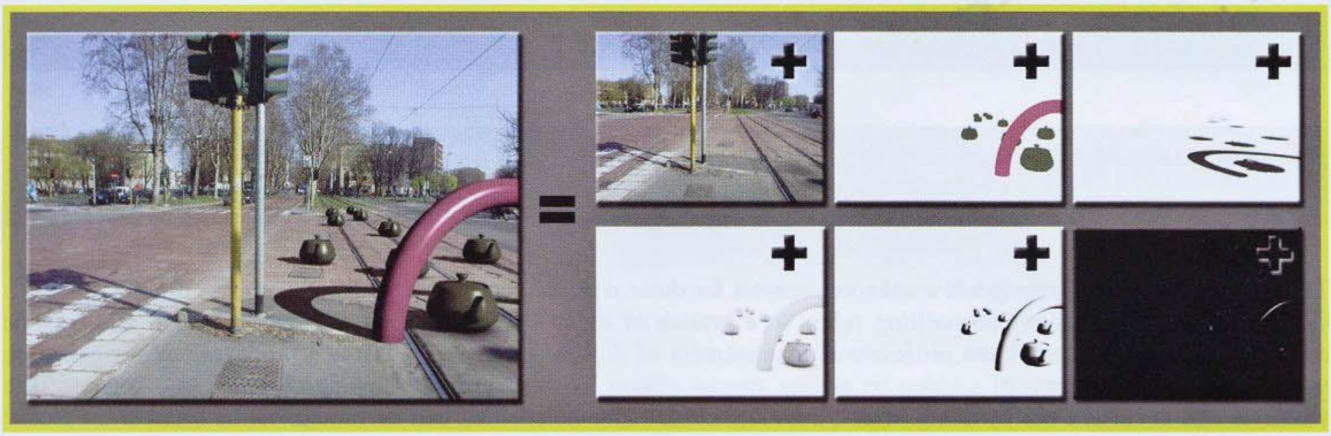
■ **Figure 7.102**
Digital Fusion, a famous Video compositing and Post production software. In the example, the green background of the main image has been substituted by a new layer representing the city. The color of the ground has been changed too.

In the case of a video or an image completely generated with Computer Graphics the situation is different compared to the compositing video of live videos. Often, once the rendering of the animated sequence is finished, it is usual not only to modify or substitute parts of objects, or the background, but to decompose the video into layers is practically a must. Due to its digital nature, it is possible to subdivide a render into layers containing the shadows, the reflections, the refractions, the specularities, the color or the Global Illumination. Once the various layers have been created, they are to be linked, like in a puzzle. The result is the original image. In this case, since many layers are available, it is possible to intervene on the shadows, the reflections and refractions without launching the render again. Simple operations with a post-production software are enough, such as Adobe After Effect, Combustion, Nuke, Fusion for correcting eventual errors in the color settings, or the shadows settings, etc…



■ **Figure 7.103**
Decomposition of an image in elements. Six layers have been created in this case.

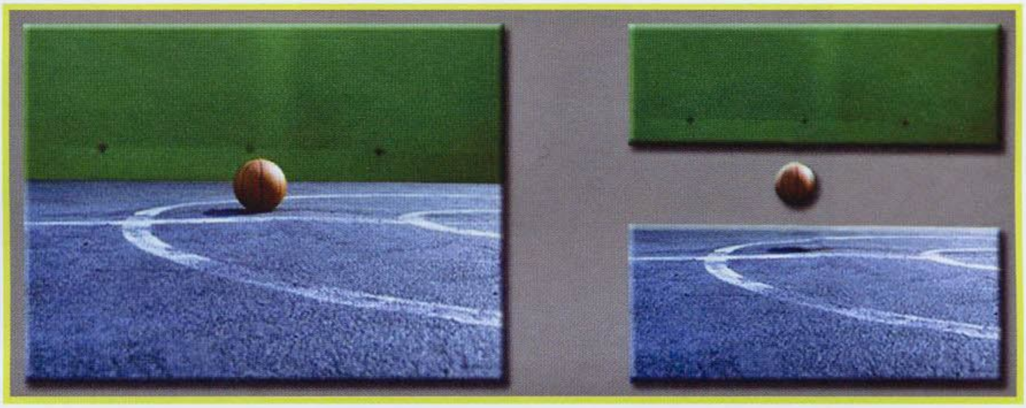
The previous layers, once linked, give the complete image back.



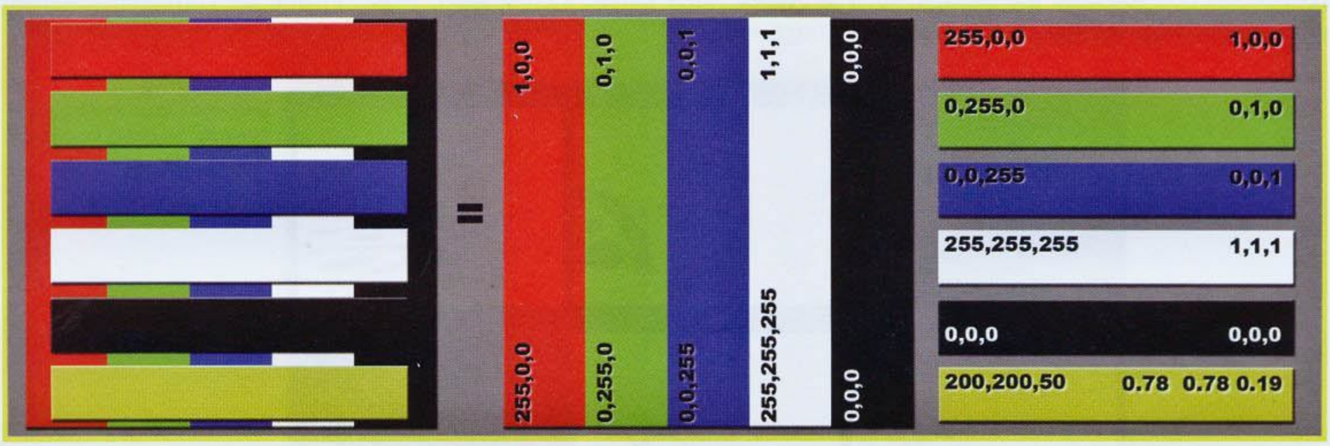
■ **Figure 7.104**
The union of all the layers allows one to generate the final render.

In live images or videos, however, it is possible to work "by layers" only with the objects, that is, by intervening as if, getting the idea from the following image, the ball is on a layer, the background on another and the basketball court on another. In total three layers would be available which, once composed, would give the final image back. With the separated layers modifying the color of the ball, the Background and the plane is simple to change.

■ **Figure 7.105**
Subdivision of the layers per object. This type of subdivision is present for videos shot live, since the subdivision of the different channels such as the specularity, the reflections, the color, the Global Illumination, etc.. is more complex, perhaps impossible.



Now we must introduce some ideas concerning the operations among different layers. Until now we have talked about "sums of layers" only, but this affirmation is not completely correct. In fact, there are not only additions. The layers can be multiplied, subtracted and divided. In the next tests the following textures are used. It is an image made up of two layers. The first one contains some Red, Green, Blue, White and Black bars, whereas the second one contains the same colors, with a gold-yellow color in addition. On the left we can see both the colors in RGB scale and in float numbers. In order to obtain the colors in float numbers it is enough to divide the R, G or B value by 255.



■ **Figure 7.106**
Image with different colors for understanding the operations between layers.

ADD (Addition)

It is possible to sum two layers by using a program such as Photoshop: it is enough to select the ***Linear Dodge*** (***Add***) mixing mode.

Figure 7.107
Sum of two layers.

The addition calculations between two colors is very simple: it is enough to sum the colors in float numbers. This way the result is obtained and, since monitors are able to reproduce just 256 colors, everything is normalized to 1.
The main observations are two:

- by summing any color to black, the result will always be the summed color.
- by summing any color to white, the result will always be white.

SUB (Subtraction)

The subtraction mode is not present in the Layer section in Photoshop. It is necessary to use the ***Add Image*** command to subtract two layers. One has to locate the two layers in such a way to put the one to be subtracted underneath the first operator (the layer with horizontal lines). Afterwards, with the "highest" layer active, the command ***Apply Image*** is to be activated. Within the tool, the layer to be subtracted is chosen, represented by the one with horizontal bars. At the end the "subtract" blending mode is chosen. The result is just one layer, not modifiable, generated by the subtraction.

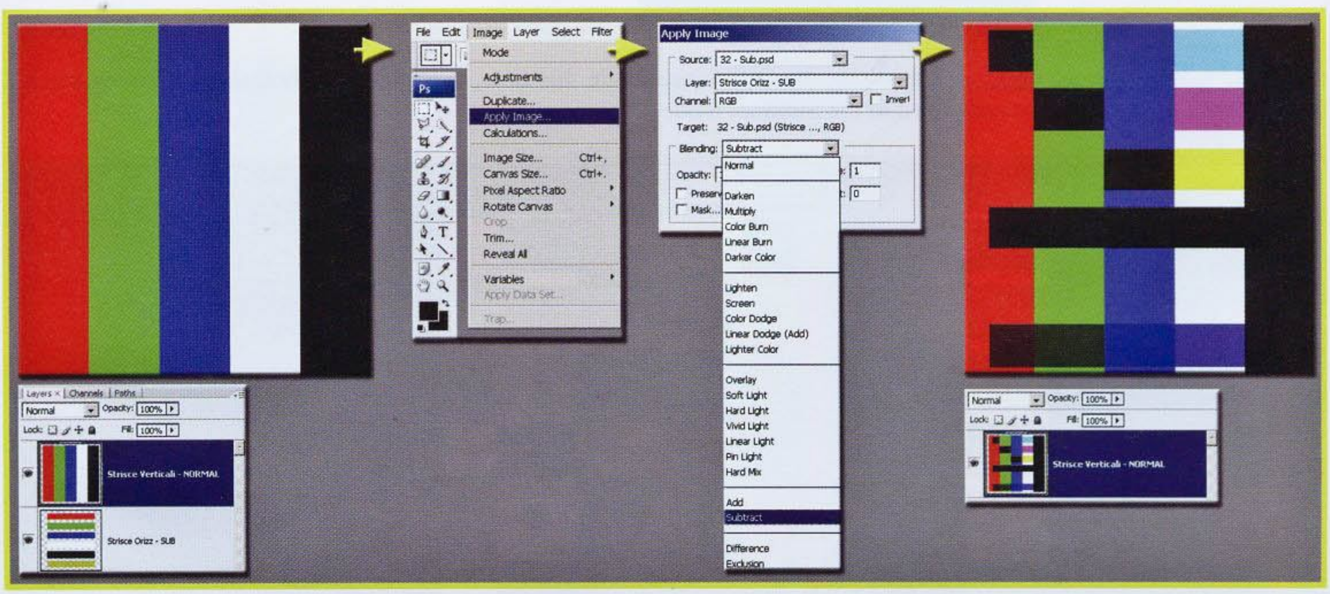
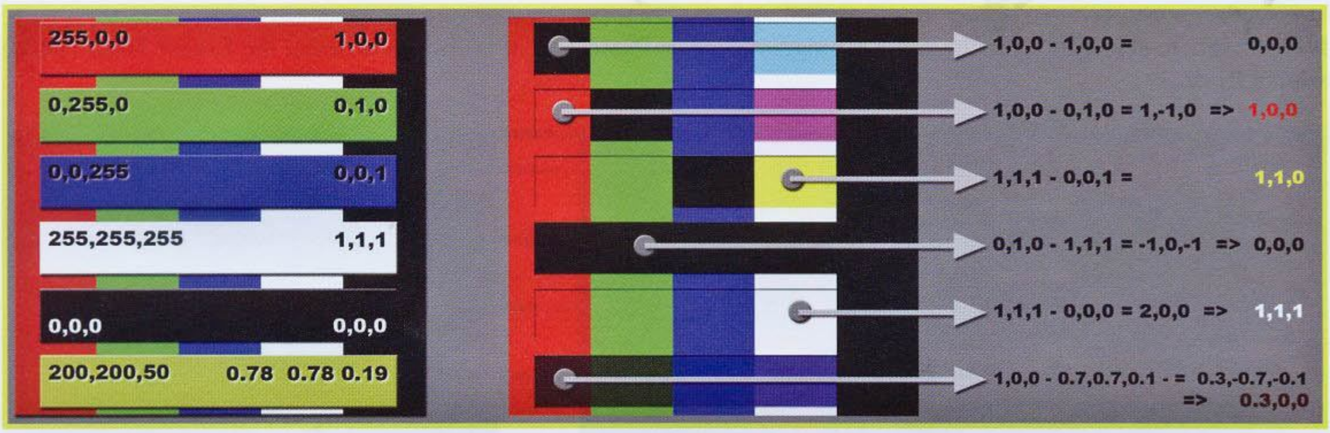


Figure 7.108
The layers subtraction inside Photoshop.

Now the numerical results of the subtraction are shown.



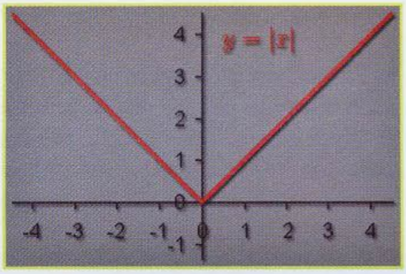
■ **Figure 7.109**
Some results obtained by the subtraction of two layers.

It is clear how the subtraction, compared to the addition, produces darker results. What mainly interests us is that:

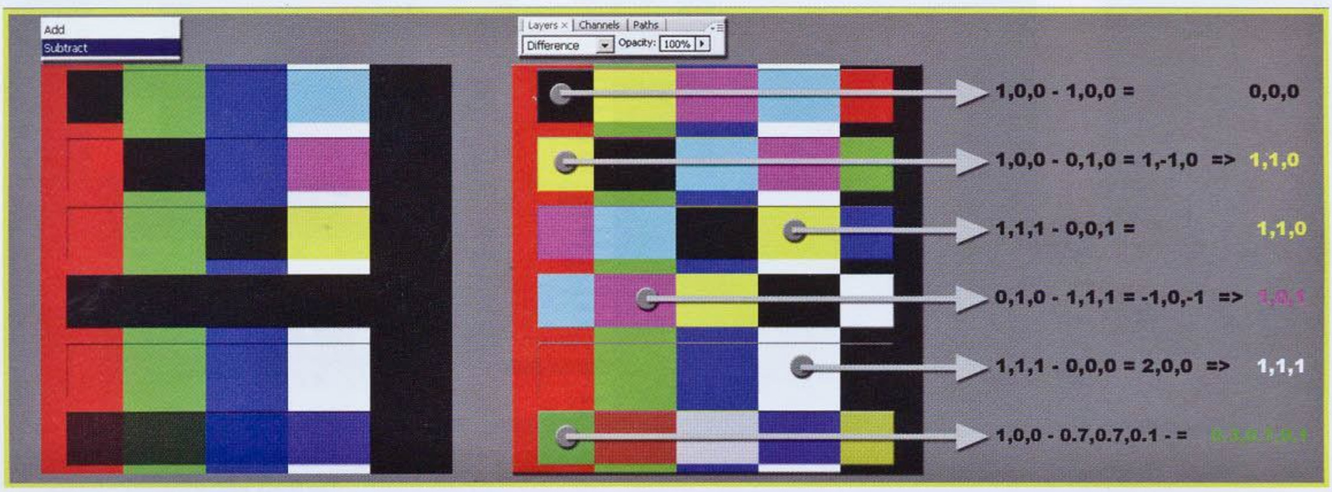
- A color subtracted to itself gives black as result.
- Any color to which black is subtracted gives back the same color.

Another type of operation exists, similar to the subtraction, called Difference. Difference differs from Subtract just for normalization: it does not normalize to 0.0 or 1.0, but the absolute value is taken. In mathematics, the absolute value of a real number is a function associating to the variable x the same number if the number is positive or zero, and the opposite value if it is negative. For example, 3 is the absolute value both of 3 and -3. 0 is the absolute value of 0 and nothing else. Given a number x, its absolute value is represented by |x|. An absolute value of a number is never lower than zero. The absolute value of a real number lower than zero is –r, that is, it is obtained by placing a – before the number.

■ **Figure 7.110**
The graphic of the "Absolute Value function".



Using the subtraction example, and applying the Difference mode this time, available via Photoshop's mixing modes, the result is slightly different compared to the Subtract mode.

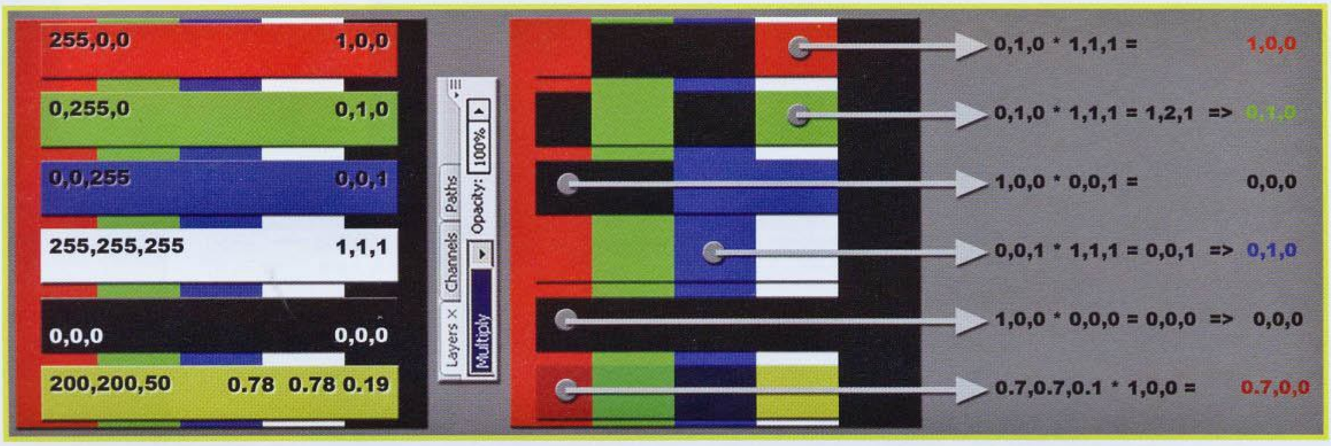


■ **Figure 7.111**
Some results obtained with the difference between two layers.

By using the Difference mode, the only way of obtaining black color from the algebraic operation is to subtract any color to itself.

MULTILPY (Multiplication)

For better understanding the meaning of "multiplication of colors" Photoshop is used again, since it is possible to fulfil the multiplication among levels through the modifications of the mixing mode.



■ **Figure 7.112**
The multiplication of layers within Photoshop.

From the analysis of the image some observations arise:

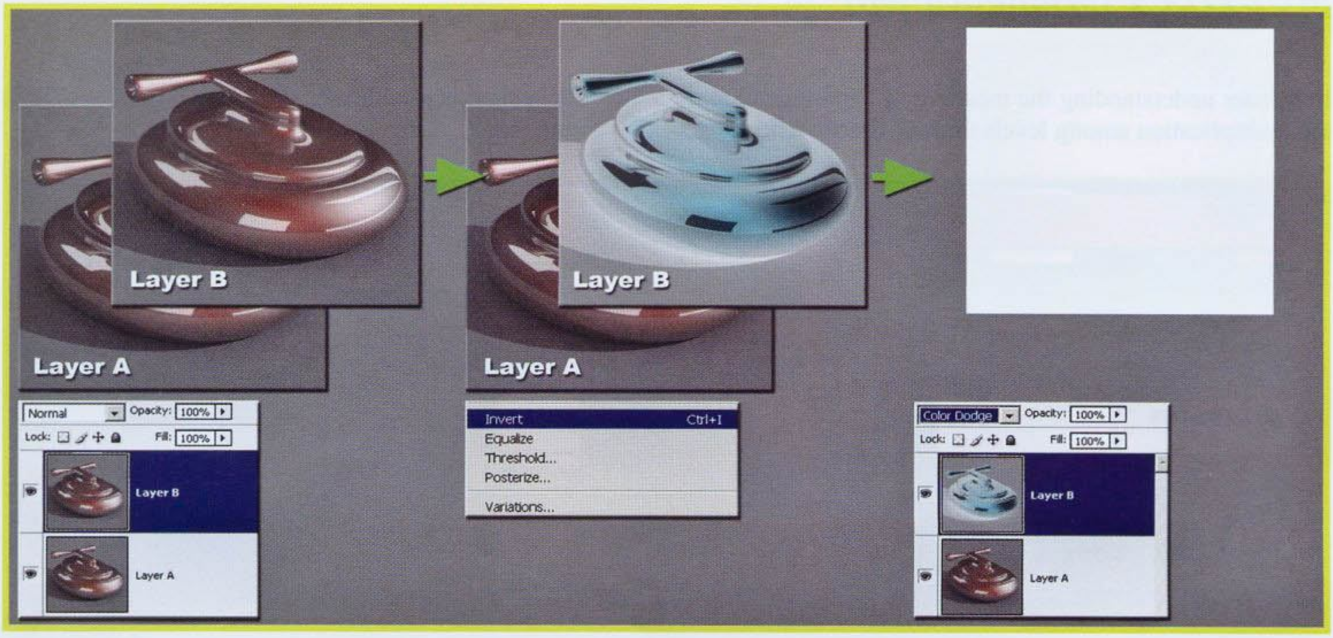
- The multiplication of a color with white gives the same color.
- The multiplication of a color with black gives black as result.

DIVIDE (Division)

Division: inverse operation to multiplication. Unfortunately Photoshop does not contain the Divide mode among its operators, which is instead found in GIMP.

■ **Figure 7.113**
Gimp and its Blending modes. Notice the presence of both Division and Subtraction.

In any case, it is possible to create the Divide mode in Photoshop as well by using the ***Color Dodge*** mode. In order to create a division of a Layer B by a Layer A it is necessary, first of all, to make the Layer B lie on top of the Layer A. Afterwards the Layer B must be inverted. Once the operation is over one just has to modify the mixing mode from Normal to Color Dodge.



■ **Figure 7.114**
Division between two levels carried out in Photoshop.

In the case described above one thing can already be noticed:

- by dividing a color by itself, the result is always white. Considering any color and dividing it by itself 1 is obtained:

[0.78, 0.78, 0.19] / [0.78, 0.78, 0.19] = [1 , 1 , 1]

In the same way, by dividing black by any color, the result is always black:

[0 , 0 , 0] / [0.78, 0.78, 0.19] = [0 , 0 , 0]

■ **Figure 7.115**
Representation of a division between levels. In the first operator black is being used.

GENERAL THEORY

What has been learnt in the previous pages is necessary for understanding the next topic: compositing. Compositing is a common job in video editing when one works on animations shot live or on a full CG. It must be said that it is possible to composite a static image too. Certainly a decomposition in layers might not be necessary in this case, but it is close to being a must in the case of animations. There are several reasons for this, but the main one is the possibility of working with greater freedom on video corrections, for example brightening the background only, or changing the color of close-up objects.
As for videos completely accomplished with CG, rendering time is known to be the main enemy of an operator. The purpose of all the techniques studied until now is obtaining the best result in the shortest time possible. In animations also, some "tricks" exist for saving time. In an animation, the Background does not change as rapidly as the foreground objects do. Imagine a play in a theatre and virtual cameras filming the performances of the actors. Ballets, dialogues are found, but the background of the stage-set is always the same. Therefore, it is clear that the render can be divided at least into two layers: the background and the actors. It would be useless to render both the background and the actors for each frame. Instead, it is more convenient to render the background with a Fly-through method and save a lot of time. The actors are to be rendered on a different layer, frame per frame, according to their dynamicity. Then everything put together for the creation of the final video.

THE BASICS OF COMPOSITING

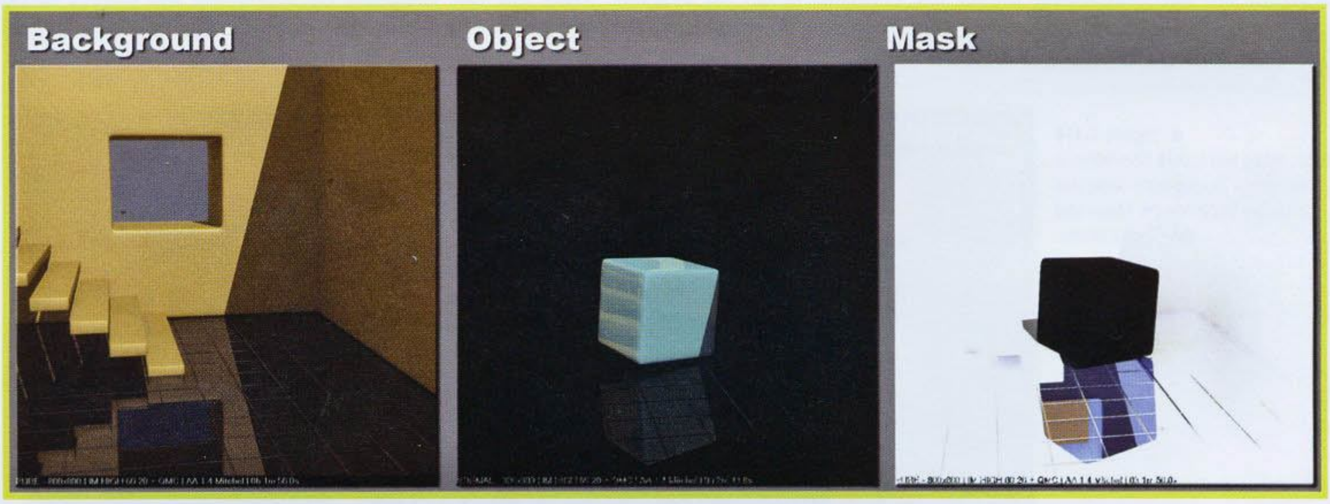
Anyone who works in the post-producing world, more precisely in the field of compositing, needs separate layers to work on. The easiest example is when two layers are available:

1. The Background.
2. The object to be composited.

In this case the layers, which should be actually two, instead become three:

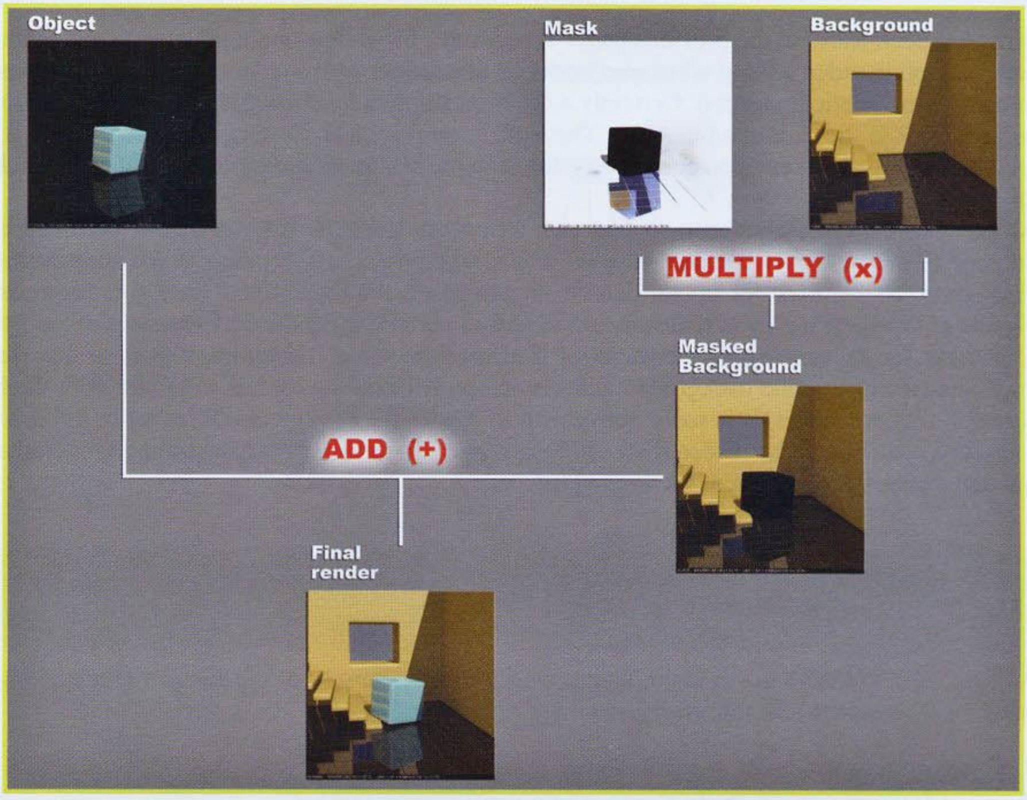
1. The Background.
2. The object to be composited.
3. The Mask.

The first two layers are quite easy to imagine. It is not the same for the third one. With this layer one determines which part of the Background layer is to be modified.



■ **Figure 7.116**
The three layers to be composited.

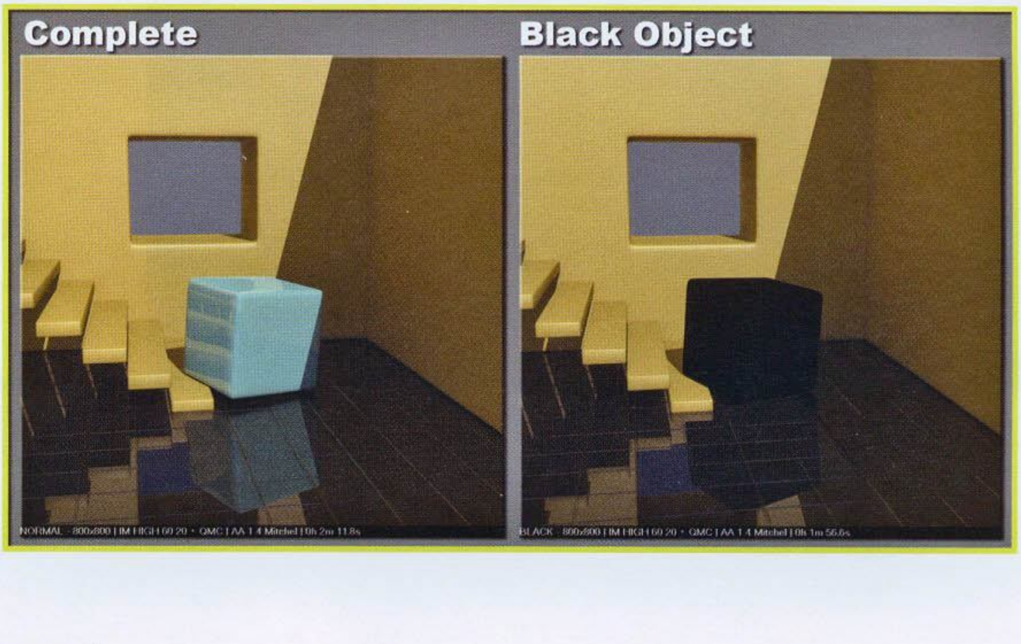
But we must understand how to "put together" these layers in order to form the final image. We have all learnt all the notions on the mathematical operations on the colors, therefore the first thing to do is to multiply the Background by the mask. The result is going to be summed to the Object. The final image is hence obtained.



■ **Figure 7.117**
Representation of the two operations to be carried out with the three layers.

Start by observing the first operation. By multiplying any color by white, the result, as seen before, is the same color. As a consequence, by multiplying the Background by the Mask, the result is the same Background, inside which a black space forms, ready to receive the Object.
The uniting operation between the Masked Background layer and the Object Layer, which is nothing more than a black image containing the object, leads to the final image. In fact, by summing any color to black, the result is the color. The two layers Object and Background have complementary portions of black. Where black is present in a layer, in the other we find the color to be added. Like in a puzzle, the two layers fit perfectly.

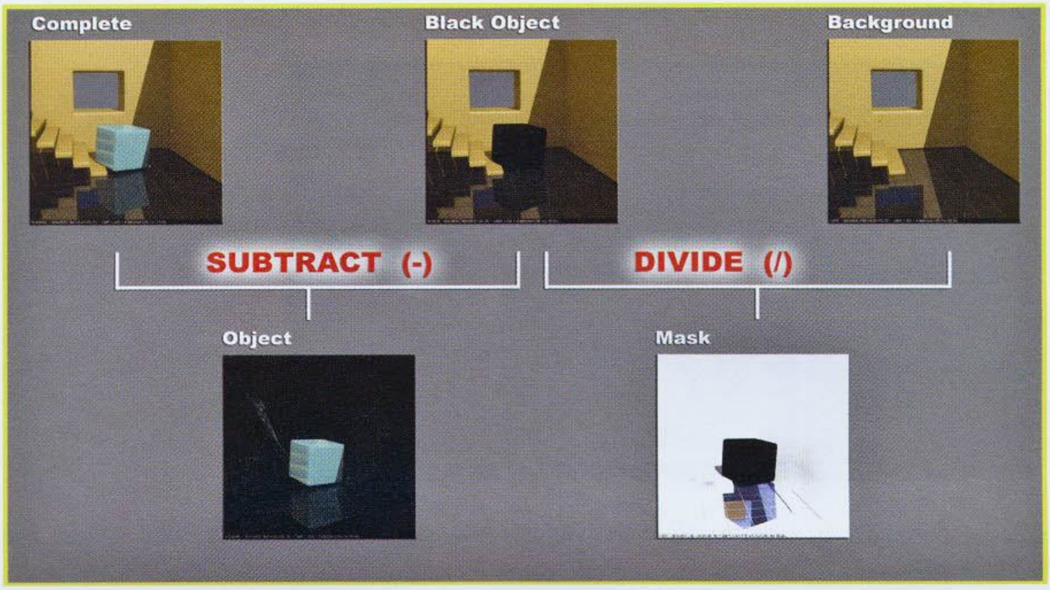
The background is quite an easy layer to derive, obtainable, for example, from a picture, a film segment or a render. As regards the two remaining layers the creation is not so evident. They are not obtained by rendering the object to be composited in some strange way, but by means of mathematical operations among layers. In particular two new layers are used, called Complete and Black object.



■ **Figure 7.118**
From these two layers and with the aid of the Background layer it is possible to obtain the Mask and the Object layers.

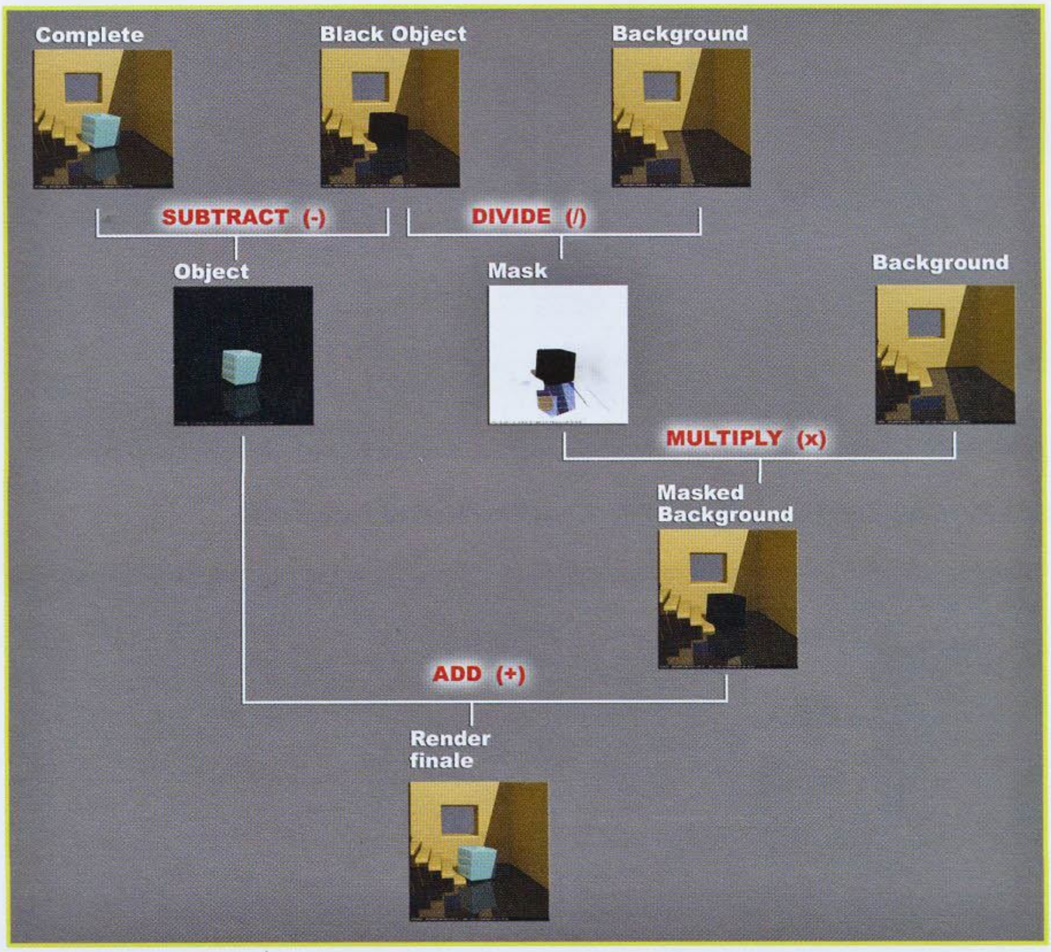
There should not be problems in obtaining the Complete layer, whereas for the black object layer, it is enough to temporarily assign to the object a completely black material.

For obtaining the Object and Mask layers previously seen, one respectively has to subtract the black object layer from the Complete layer, instead for the Mask one has to divide the black object layer by the Background layer.



■ **Figure 7.119**
By using some simple renders as operators it is possible to obtain two fundamental layers for compositing: the Object layer and the Mask layer.

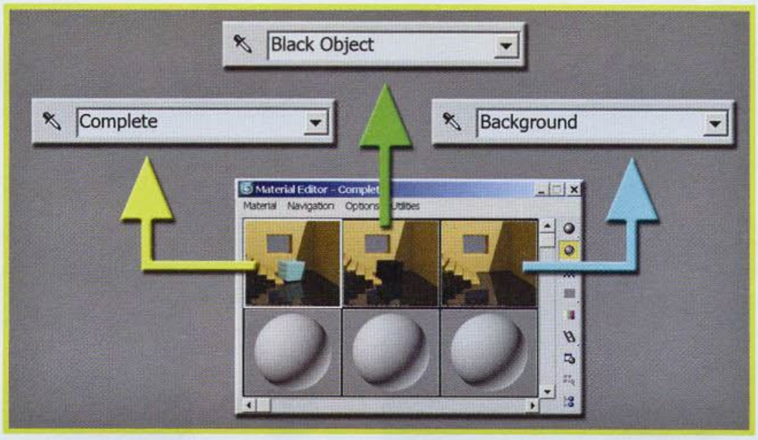
As far as subtraction is concerned, since the two operators are the same in everything except for the object, the result obviously is the same mesh. In fact, by subtracting a color from itself, the result is black RGB = [0, 0, 0]. Vice versa, by subtracting black from a color, the result is the same color, that is the color of the cube with all its reflections. As for the Divide operation, by dividing a color by itself, the result is 1, while by dividing black by any color the result is always black.
Now it is possible to globally represent all the layers and the operations achieved with Photoshop for the composition of the final render.



■ **Figure 7.120**
The decomposition on layers of a render. Only three renders are used.

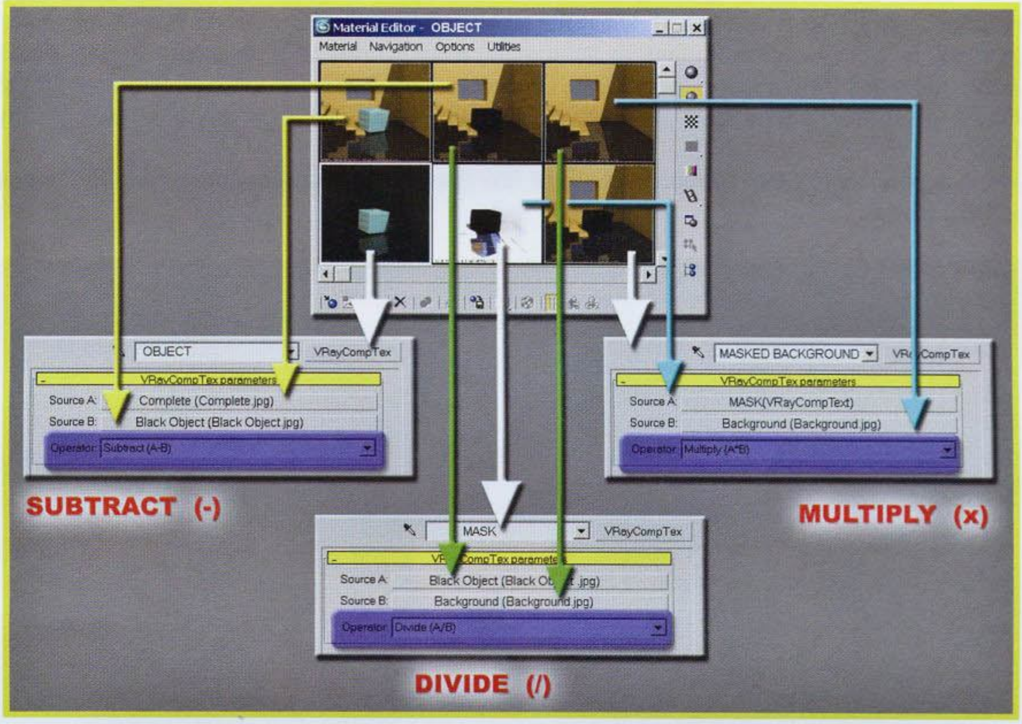
So far Photoshop has always been used for accomplishing all the operations on the levels. Well, *VRay* offers, thanks to its ***VRayCompText*** map, a little management tool for the layers, born for aiding compositing. Thanks to the possibility of modifying the operations on the layers and of creating materials nested into each other, it is possible to carry out what has been done so far in Photoshop, inside 3ds Max.

First of all the Bitmaps of the three main renders, Complete, Black object and Background, have to be loaded by inserting them in the Material Editor.



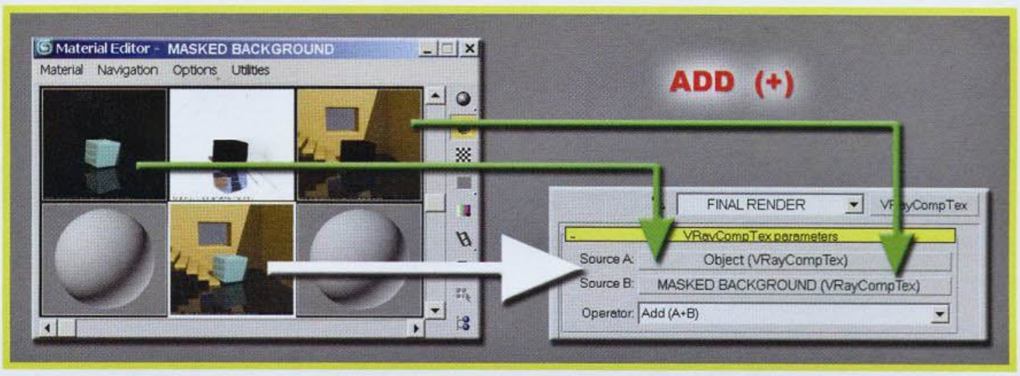
■ **Figure 7.121**
The three main renders, ready to be used inside the ***VRayCompText***.

Now it is necessary to compose the maps, in order to fulfil the Object, Mask and Masked Background layers. The ***VRayCompText*** is used for this reason.



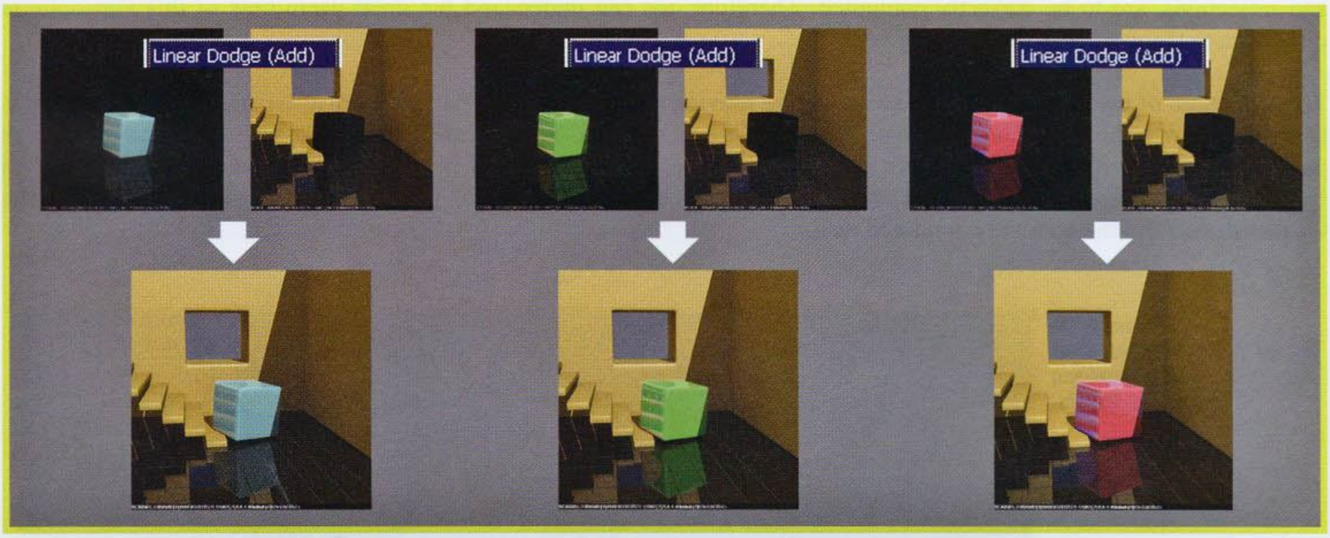
■ **Figure 7.122**
Three ***VRayCompText*** maps with the respective sources.

Now one just has to sum the ***VRayCompText*** Object and the ***VRayCompText*** Masked Background.



■ **Figure 7.123**
The final render.

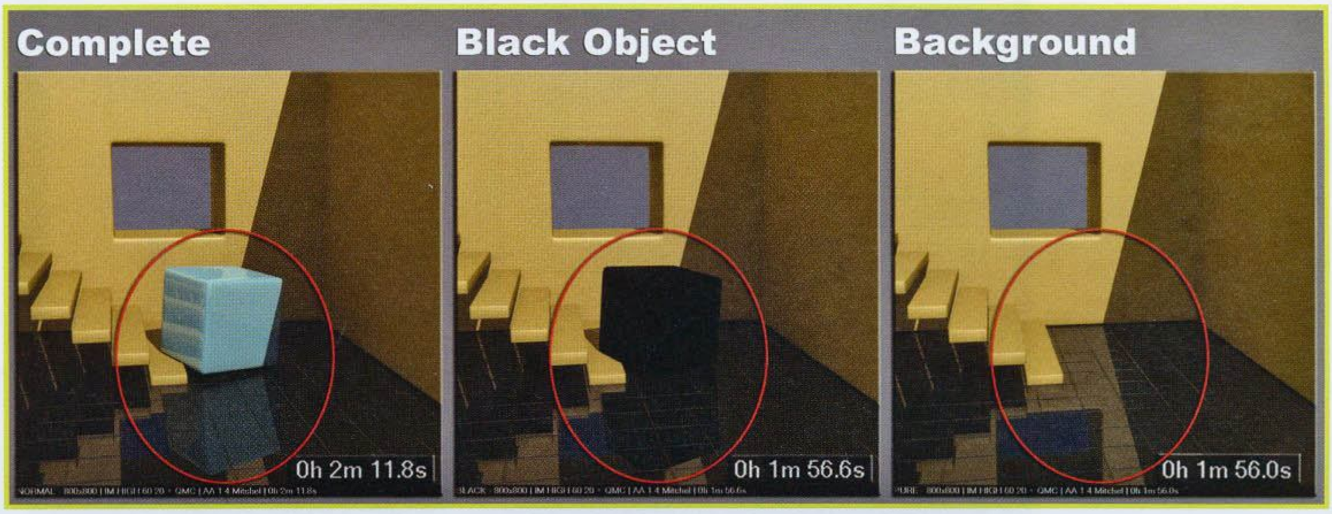
As planned, the final render has been decomposed into two layers, then after which are put together. This way it has been possible to intervene with Photoshop's Hue/Saturation command changing the box color in a very simple way, without having to select anything. The selection is, in fact, the whole layer.



■ **Figure 7.124**
Change of the color of the Object layer by means of Photoshop's Hue/Saturation command, after put together with the Masked Background layer in Linear Dodge mode. The new color perfectly fits with the rest of the image.

PARAMETERS

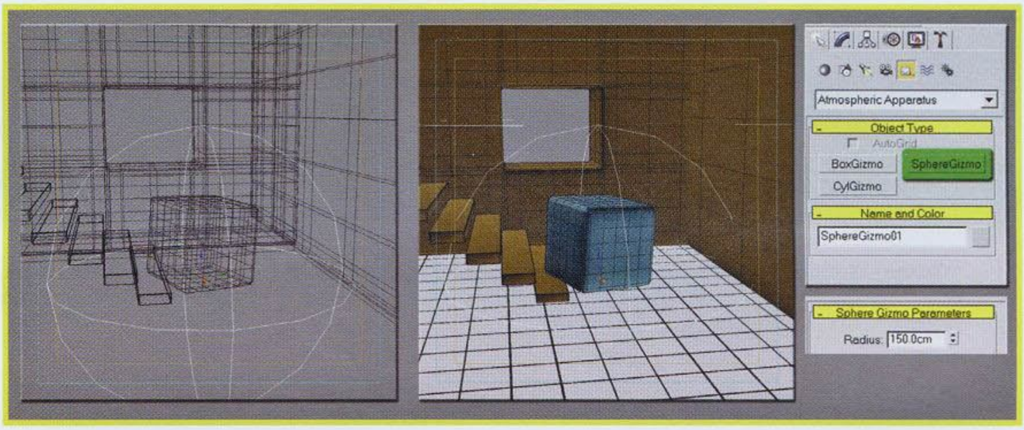
The advantage of the method just described for compositing an image, is that the user must not take care of setting the Matte/Shadows materials or risking to forget reflections, refractions or other effects. But there is a little problem. The previous method could require long rendering times when animations are involved. By observing the three starting renders one may notice that many parts of them are equal. The only differences concern the object to be composited. The walls are unchanged in all the images, like the stairs and some parts of the flooring.



■ **Figure 7.125**
Inside the red circle the area with most of the differences is present. Notice that the floor is also part of it, due to the fact it reflects the box.

Having to render the same areas for all the three images is obviously useless. In an animation, where frames can be a few thousand, one can see how much time is wasted. This is where the ***VRaySphereFade*** comes into play. Its main feature is that it allows one to render just one part of the scene, which is, in this case, the area bounded by the red circle near the blue box. The rest of the image instead remains grey, while in the rendered areas all the effects, GI, refractions and reflections, are to be correctly computed. Therefore one can understand the great usefulness of this atmospheric effect due to the great deal of time it saves.

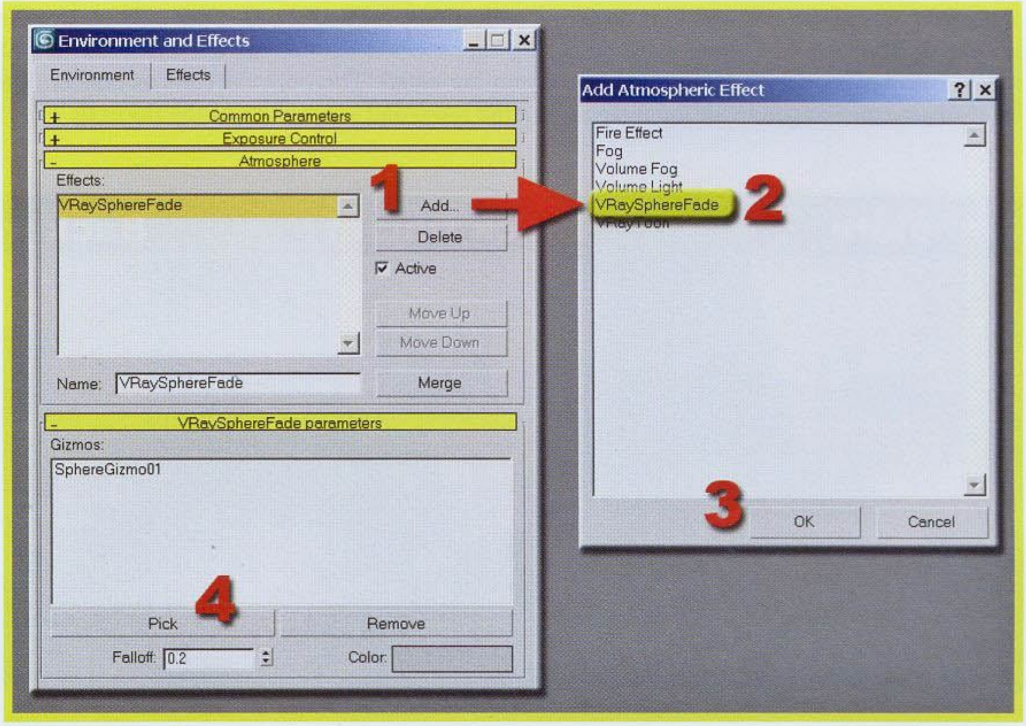
The procedure of use is very simple. First of all the moving object is found. Then a spherical Gizmo is created, which is to be "linked" to the object on the move. This way the helper is to be connected to the movements of the mesh. The possibility of animating it manually is not excluded. The gizmo is located in the command panel, under Create -> Helpers -> Atmospheric Apparatus. The area within the Gizmo is the one to be rendered: the higher the size of the Gizmo, the greater the area to be rendered is.



■ **Figure 7.126**
With the ***VRaySphereFade*** effect it is possible to use only spherical Gizmos.

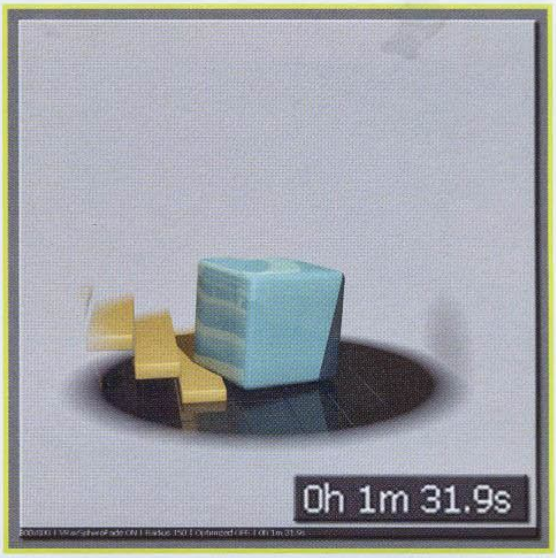
The Gizmo, with a radius of 150 cm, has been perfectly aligned to the pivot of the box, in such a way as to contain the entire geometry, a little portion of the floor, stairs and the walls.

Now the ***VRaySphereFade*** has to be added among the atmospheric effects, and the Gizmo just created has to be put inside it. It is possible to insert several Gizmos as well.



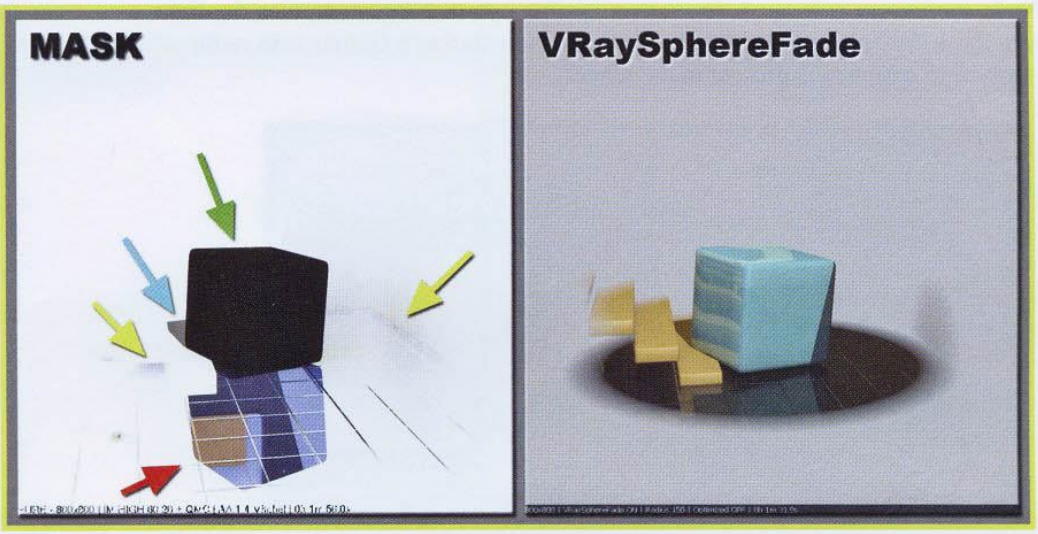
■ **Figure 7.127**
Procedure for the ***VRaySphereFade*** effect activation and introduction of the Gizmo.

Now all that is left to be done is render.



■ **Figure 7.128**
The *VRaySphereFade* atmospheric effect in action.

The Gizmo is perfectly aligned to the base of the box and everything within it is correctly rendered. All the rest is gray light. At this point a brief consideration is needed. One can tell which are the portions of the image to be inserted in the Gizmo by observing the Mask layer. Because of the reflection of the box on the ground, the size of the Gizmo in the previous image is not enough, since it was able to contain only the box, but not its complete reflection on the floor.



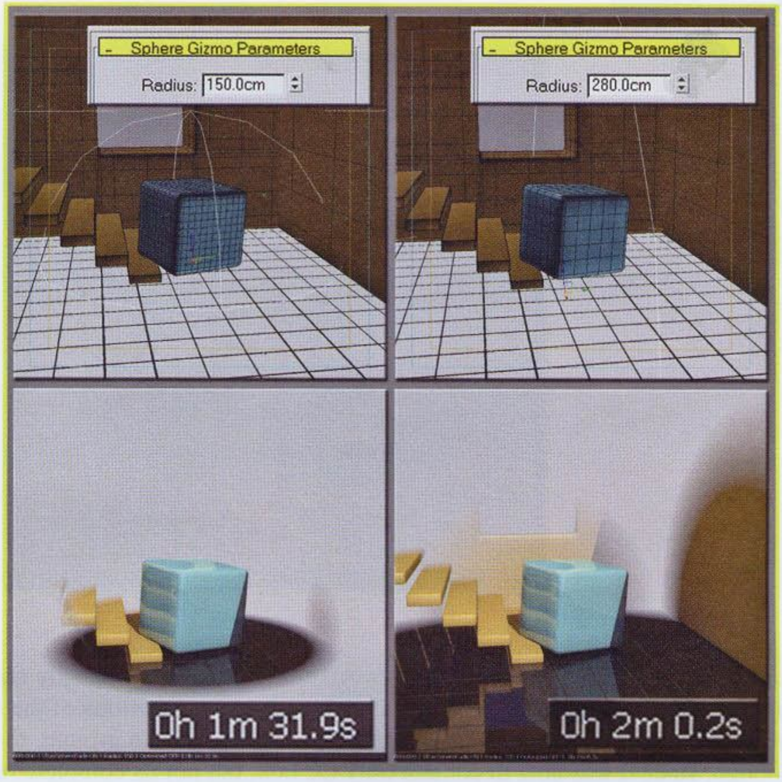
■ **Figure 7.129**
The mask and the rendering with the *VRaySphereFade* effect.

In order to obtain the best result, everything black in the Mask layer should be inserted in the Gizmo;

. Red arrow: the reflection of the object.
. Yellow arrow: the GI generated by the object.
. Blue arrow: the shadow generated by the object.
. Green arrow: the object.

One just has to reposition the Gizmo in such a way as to contain as many elements as possible. But just the displacement of the Gizmo is not enough. The radius has been increased too. In such a way the whole reflection on the floor has been covered. Now the area within the ***VRaySphereFade*** covers almost 50% of the same image. The rendering time is affected by this: it has passed from 1 min. and 30 s. to 2 min.

■ **Figure 7.130**
The Gizmo, beyond having a major radius, has moved along the X-axis, approaching the camera. This has allowed the coverage of the reflection on the floor.

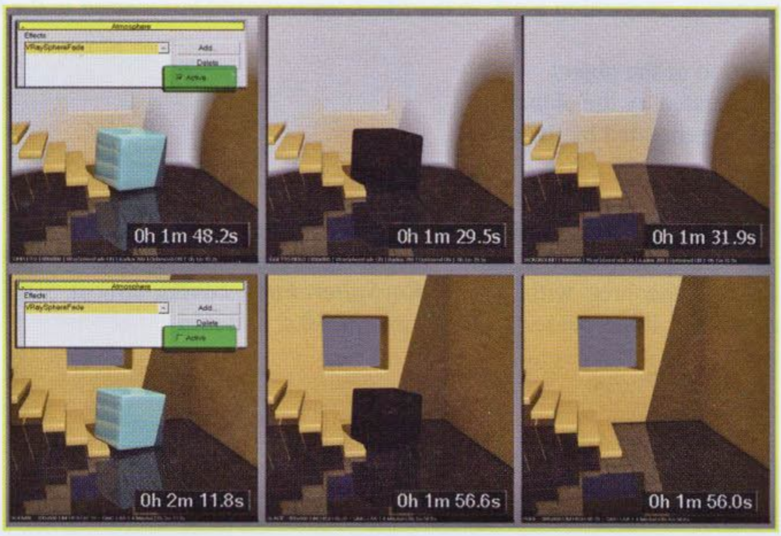


Now the areas involved by the object are all inside the Gizmo. In the Renderer panel, in the rollout ***VRay: System*** the so called ***Optimized Atmospheric evaluation*** is located. By activating this option *VRay* computes the light-grey areas in the image faster, hidden from the ***VRaySphereFade*** effect. For the demonstration a Gizmo with radius 150 cm is used. In such a way a wide grey surface to optimize is present in the render.

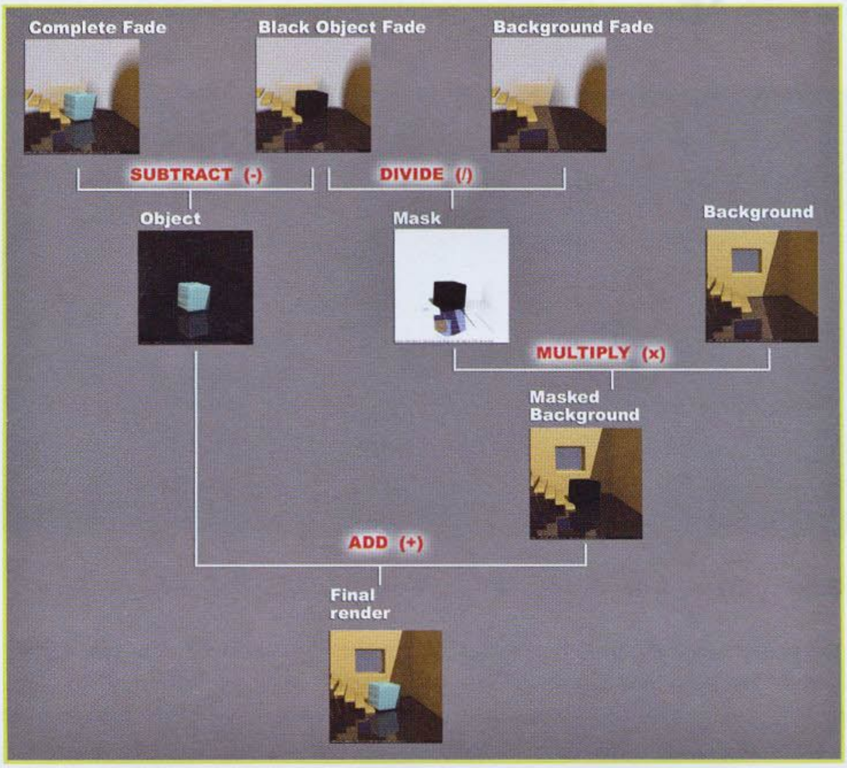
■ **Figure 7.131**
Notice the difference in rendering time between the two tests!

Now one just has to render the three basic layers: Complete Fade, Black object Fade and Background Fade.

■ **Figure 7.132**
The three main layers. Notice the difference in the rendering time without the ***VRaySphereFade*** effect.

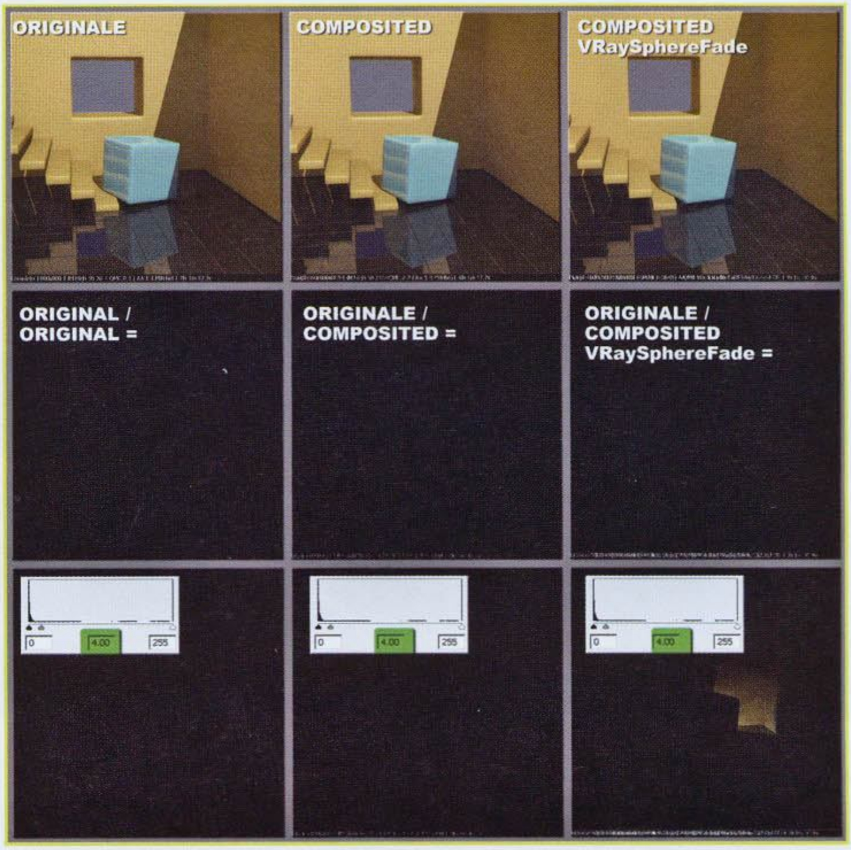


Now we shall carry out the composition of the three layers inside Photoshop with the scheme previously used.



■ **Figure 7.133**
The composition of the final render via the four layers. Three of them are achieved through the employment of the ***VRaySphereFade*** effect.

From the scheme one can observe how the subtraction of black Object Fade layer to the Complete Fade layer produces the Object layer, same to the one obtained without using the ***VRaySphereFade***. The big advantage is rendering time. Same thing for the Mask layer. Unfortunately, in this case, it has been necessary to render a fourth layer, the Background layer. This has been necessary to fill the space left by the light-grey color in different layers in which the ***VRaySphereFade*** effect has been used.
Now one just has to compare the methods just considered, that is the methods with and without the employment of the ***VRaySphereFade*** effect, with the Complete render obtained without compositing, generated directly from the output rendering.



■ **Figure 7.134**
Differences between the original render and the two methods accomplished by means of the compositing. In the first line the renders are present, while in the second there are the results obtained by the difference between the layers. The differences are minimal and it has been necessary to change the level of the average colors from 1 to 4 through the Levels parameter of Photoshop to actually notice them.

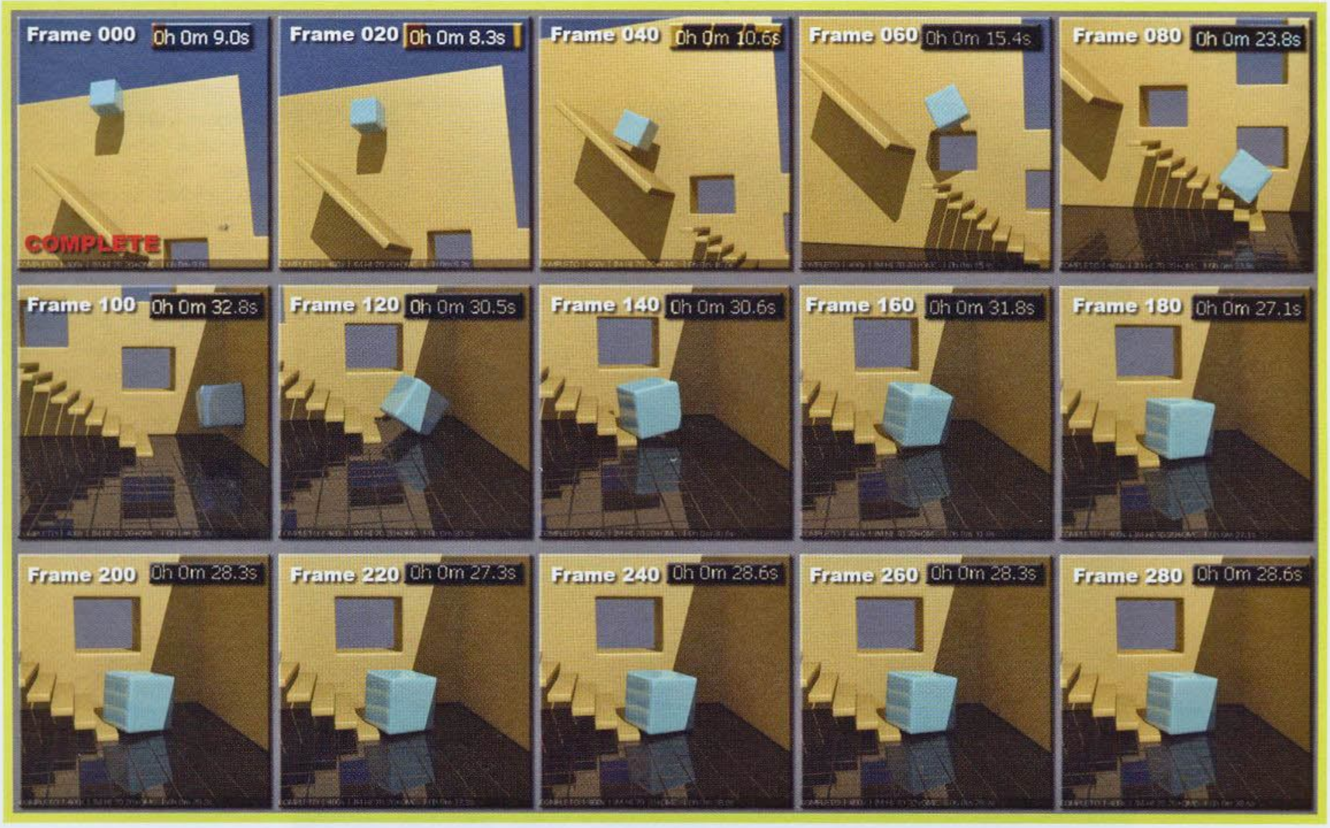
The differences between the render composited in layers without the ***VRaySphereFade*** and the non-composited layer are none. With the use of the ***VRaySphereFade*** some little problems arise. They might have been foreseeable before the compositing. With the effect active, the Gizmo, in fact, was not able to cover the angle formed by the walls behind the blue box.

■ **Figure 7.135**
The angle not included inside the Gizmo.

Even if hardly noticeable, the presence of the blue box generates, in the angle between the two walls, a certain shadow, obviously not present in the Background layer. When the Mask layer is summed to the Background via ***VRaySphereFade***, the zone not covered by the Gizmo is lacking. It would have been necessary to increase by few centimetres its radius.

In the previous exercises the employment of the ***VRaySphereFade*** for static images only has been analyzed. It is possible to use it successfully in animations as well. Or better still, the ***VRaySphereFade*** is successful most of all in these kind of computing.

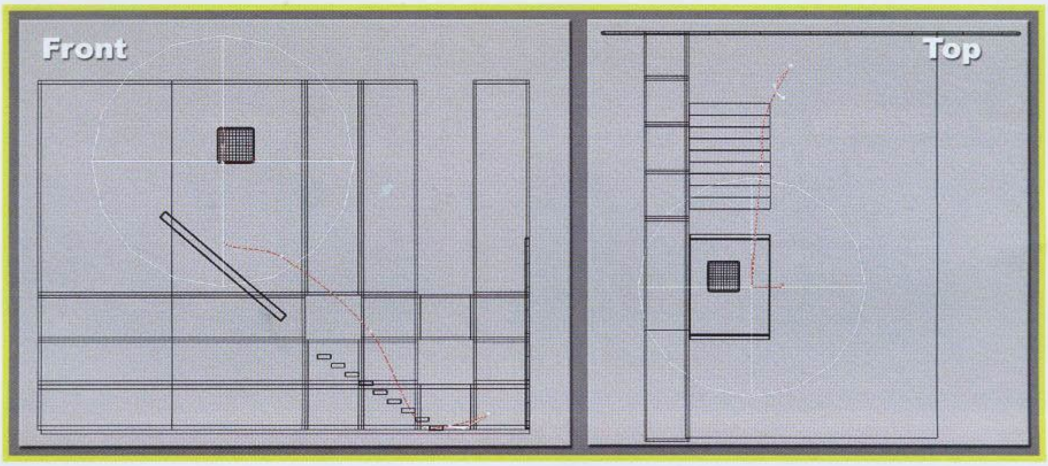
A simple animation is to be used, created by making the blue box previously used fall from above.



■ **Figure 7.136**
The animation contains 300 frames. In the previous examples the image has been taken at the 300th frame, that is the last one.

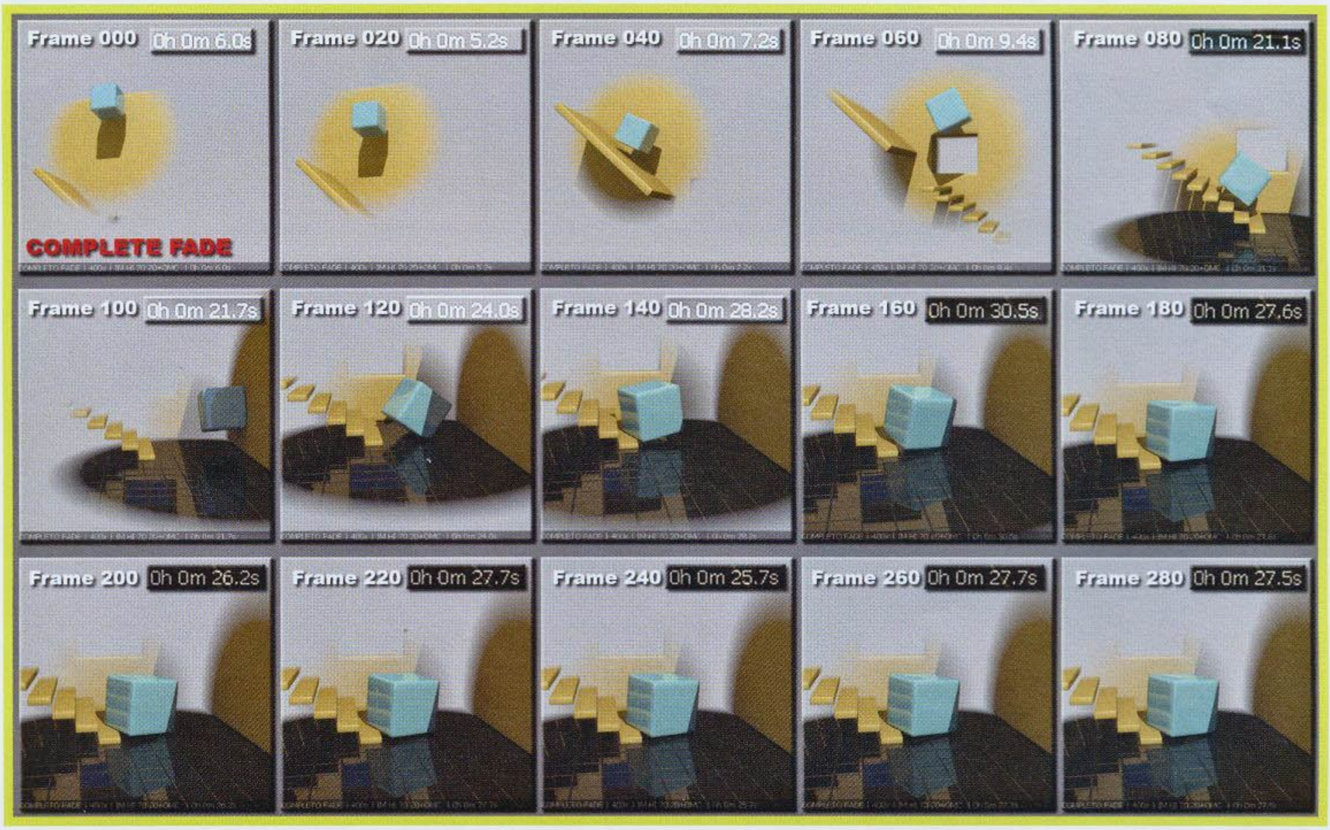
In the previous image the blue box falls on an inclined plane first, and then along the stairs. Afterwards it bounces on the wall and it stops poised on the last step. The sequence and the relative rendering times are obtained by rendering each frame in one step only, with the ***IM+QMC*** method. The ***Irradiance Map*** has been computed for each frame because of the presence of the cube moving continuously.

First of all the Gizmo has been manually positioned near the box. Afterwards it has been animated and located in such a way to cover always the object during all its animation. This has been possible just by animating it manually, each time putting it in the exact point needed.



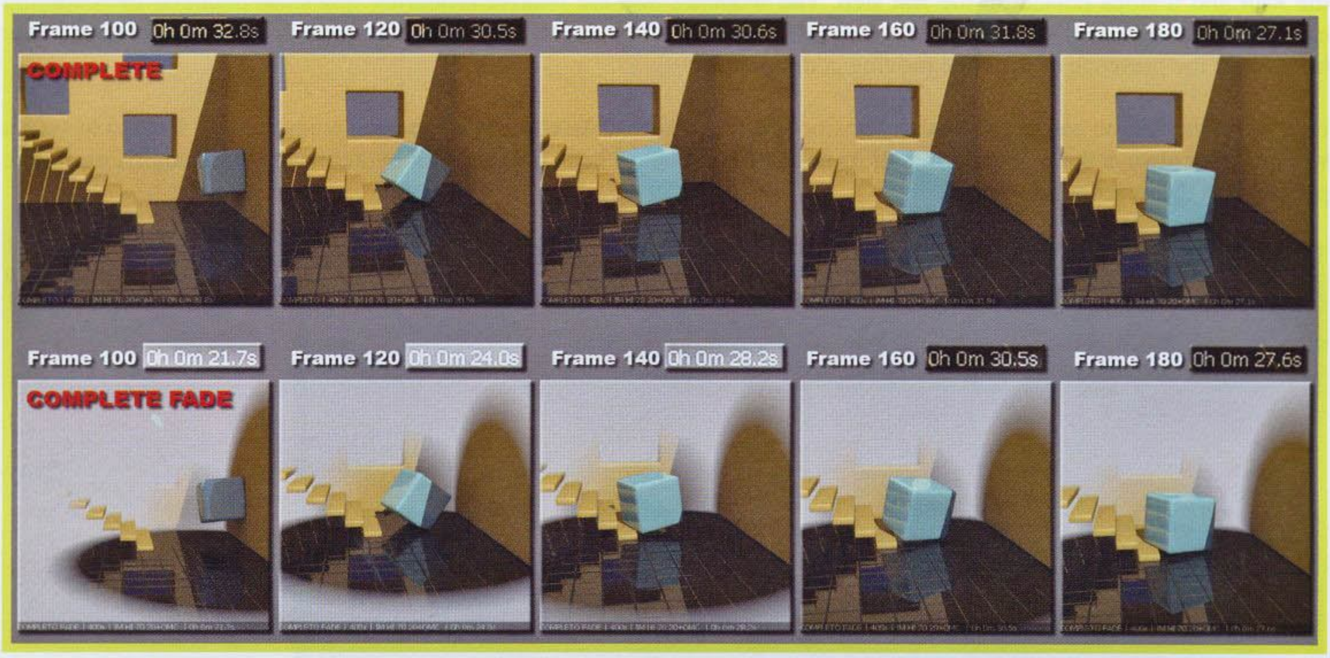
■ **Figure 7.137**
The trajectory of the animation of the Gizmo.

At this point the animation has been computed four times, in order to create layers useful for the compositing. As far as the Complete Fade and Black object Fade layers are concerned, the method used is the same as the one used on the previous animation, that is the ***IM+QMC*** method with the computing of the ***IM*** for each frame because of the movement of the object. The method used for the Background and Background Fade layers is the ***IM+QMC*** with the ***IM*** computed for ***Fly-through*** animations, because nothing was moving. Then one proceeded by computing the ***IM*** every 10 frames in ***Incremental add to current map*** mode for the Background layer, the ***IM*** was saved and reused for computing each frame. The same ***IM*** was reused to compute the Background Fade layer. In the last case, in fact, the only difference regarding the Background layer is the presence of the ***VRaySphereFade***.



■ **Figure 7.138**
The Complete Fade layer.

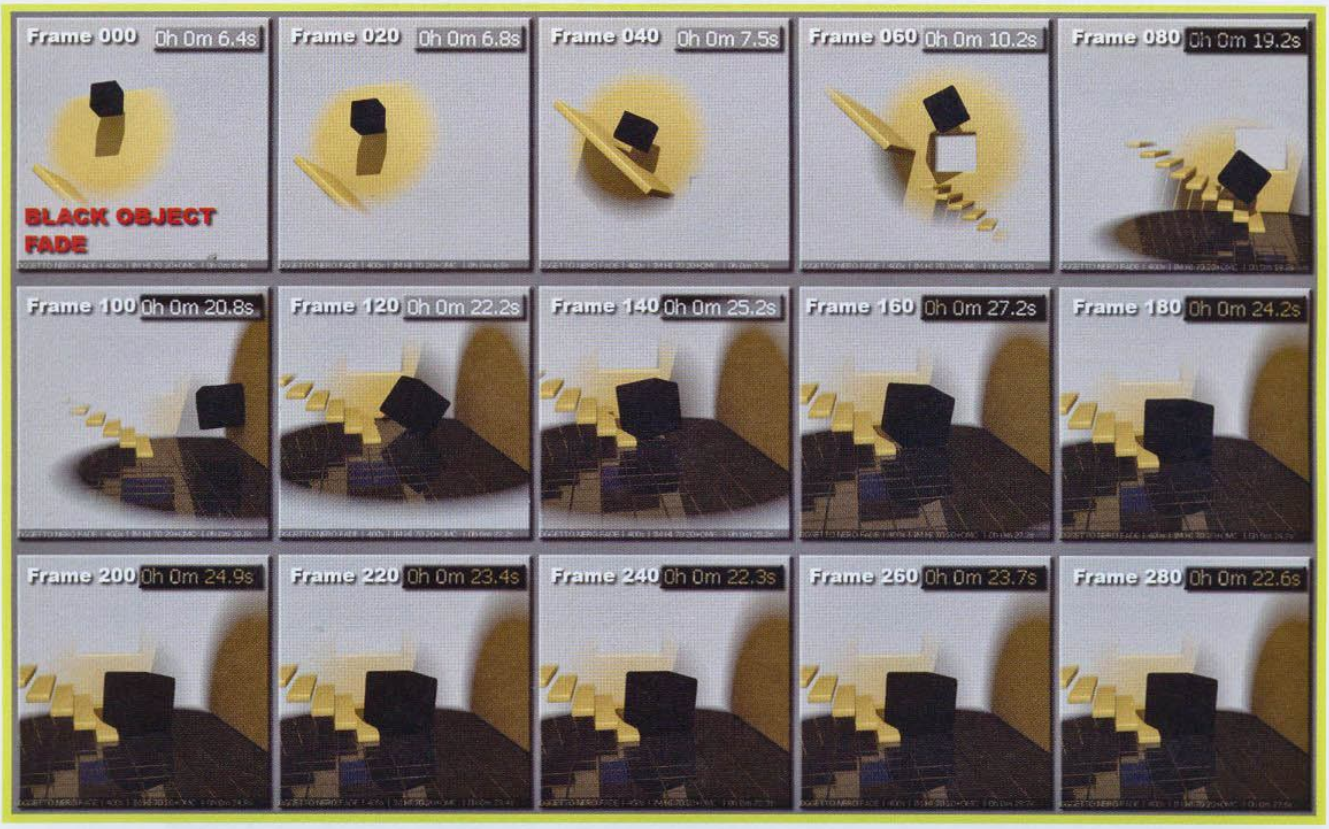
Now compare the two layers: Complete and Complete Fade.



■ **Figure 7.139**
Comparison between the two layers Complete and Complete Fade. Notice the rendering times.

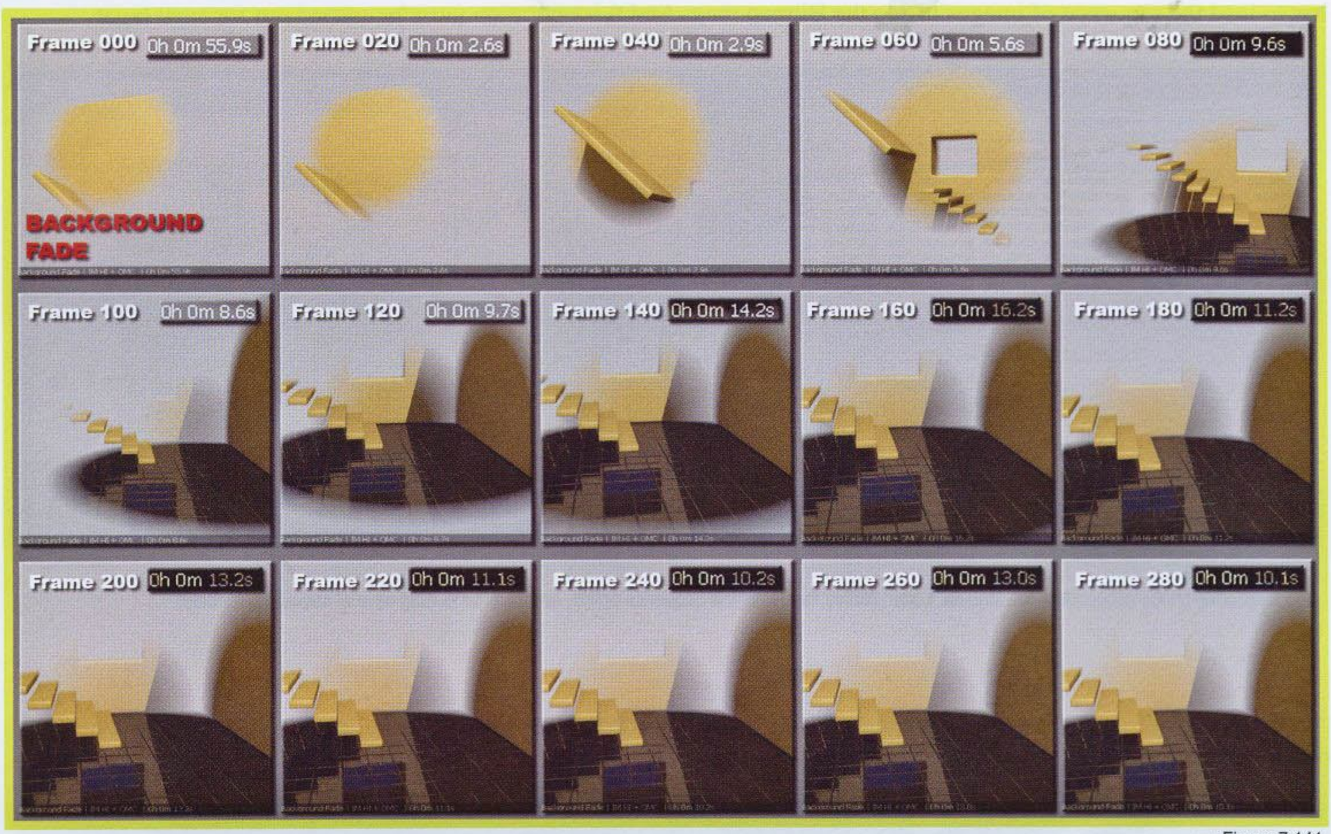
As has been already noticed, for correctly rendering reflections on the floor it has been necessary to use a very big Gizmo compared to the box. This has affected rendering times greatly, since the most complex part of the scene is the floor, present both with and without the ***VRaySphereFade*** effect. The more the Gizmo overlaps the floor, the more the rendering times of the Complete and the Complete Fade layers tend to be equal.

At this point the Black Object Fade Layer is to be rendered.



■ **Figure 7.140**
Black Object Fade layer.

And finally the two last layers, Background Fade and Background.

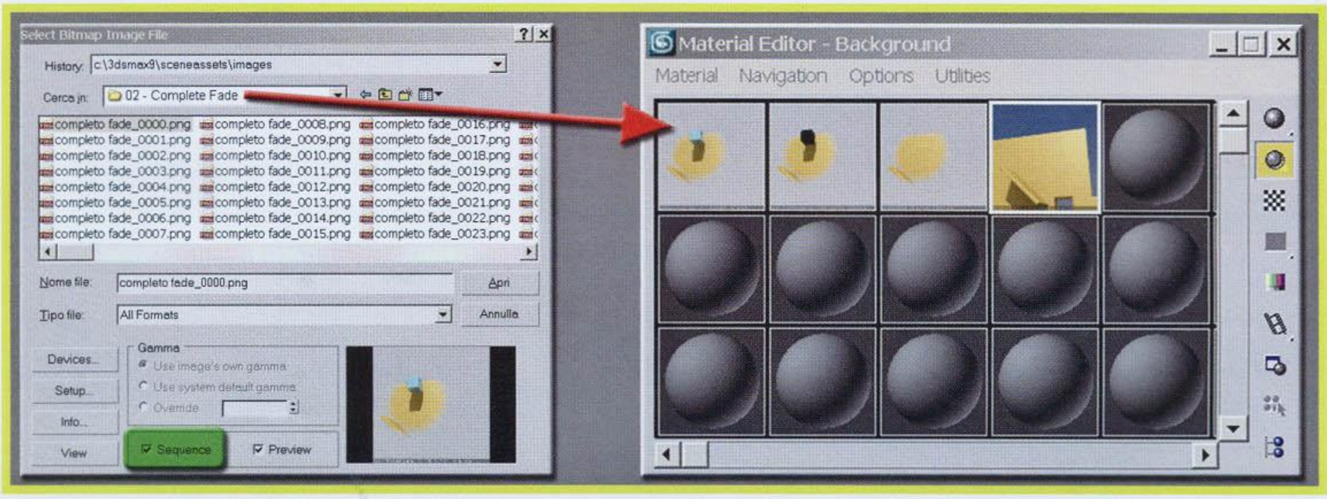


■ Figure 7.141
The Background Fade layer.

In this case rendering times are definitely lower than the previous ones, since the ***IM*** has been computed before. In this case, in fact, the rendering time includes only the final computing with the application of ***antialiasing***, colors, reflections and refractions; the GI is not included. Same thing for the Background.

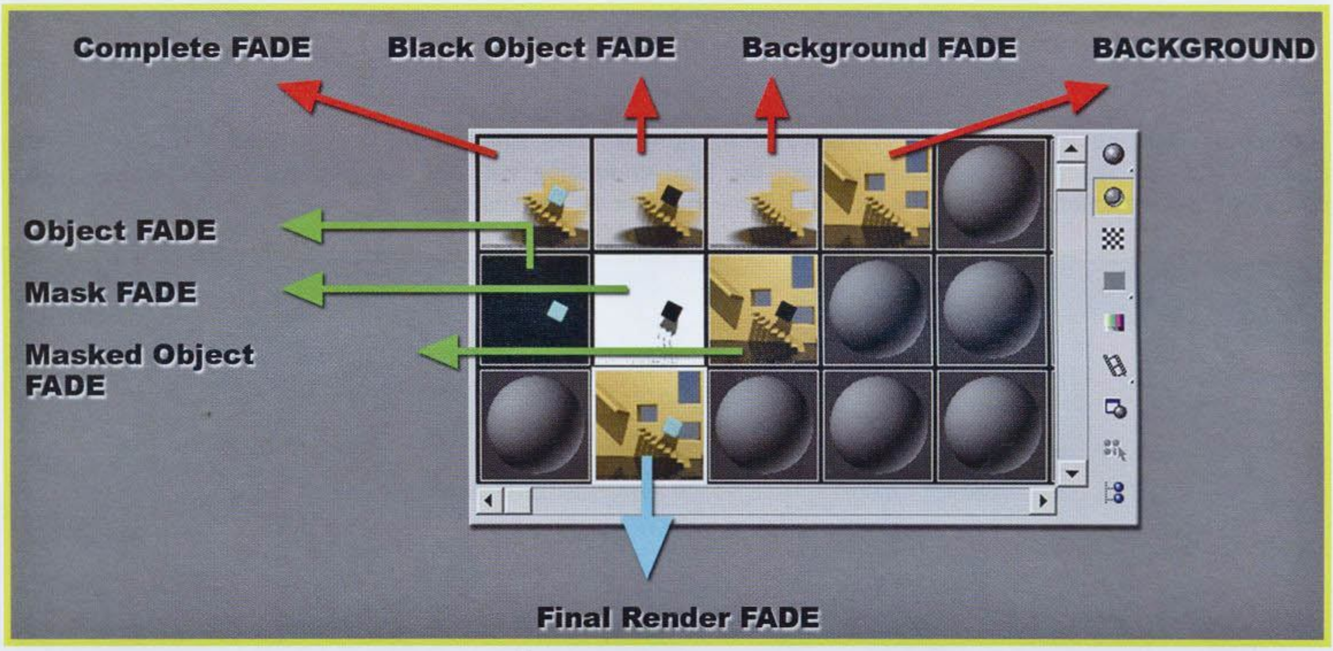
■ **Figure 7.142**
The Background layer. The first frame has high times due to problems on the net.

At this point, it is enough to composite the four layers, with ones favourite video editing program or inside 3ds Max, to obtaining the final Render. Using the layers inside 3ds Max for accomplishing a video is very simple. First of all one has to load the sequence of images inside the Bitmap map. And this operation must be carried out for each layer.



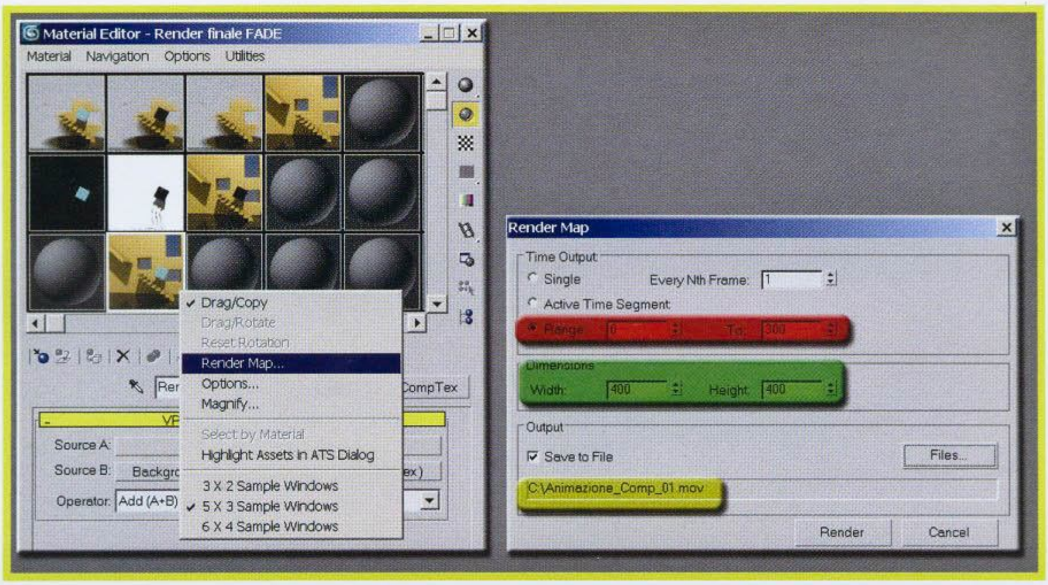
■ **Figure 7.143**
Inclusion of an image sequence in the Material Editor. Notice the use of the Sequence option, essential for loading several sequential renders.

At this point the layers must be composited. The material ***VRayCompText*** is to be used, like we described in the previous tutorial.



■ **Figure 7.144**
The Material Editor with all the layers for the compositing.

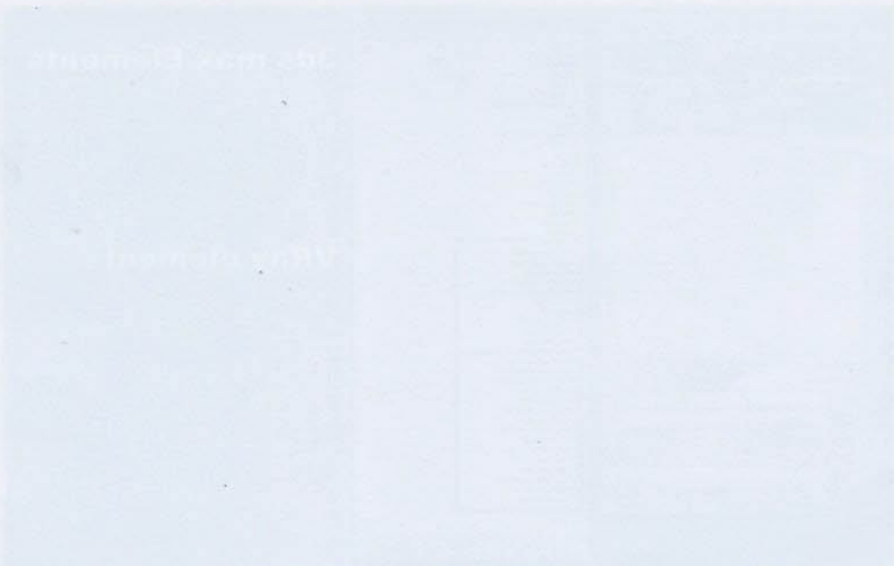
By clicking the right button on the Final render Fade layer the command Render Map appears among the other ones. A new window appears, with which it is possible to determine the exportation parameters of the video. The range, that is, the number of frames the animation is made up of, the resolution, in this case 400x400 pixels, and the output format are highlighted in the image. The file .mov Quicktime has been chosen for its practicality. The tutorial is finished. Congratulations!



■ **Figure 7.145**
It is possible to generate a film-segment directly from the Material Editor.

The Render Elements system is an integrated module of 3ds Max, allowing one to separate the information present in a render into single images. VRay just explains some of its features. A rendering is composed of objects, shadows, reflections, refractions, etc. With the Render Elements it is possible to create a separate render for each one of the listed actions. With appropriate software it is possible to use these layers for various purposes, like touching the final render by using the separated renders.

The main purpose of the subdivision of layers into channels is giving the possibility of modifying its properties in post-production. For example, there is no need to re-render a render if shadows are too dark. One just has to modify the "Shadows" layers in post-production with a software like Adobe After Effects or Autodesk Combustion in order to obtain the changes needed, saving in such a way a lot of time.
The Render Elements panel is found in the Render scene tool, in the Render Elements card. By clicking on the Add button it is possible to insert the desired layers. Notice how specific VRay channels are to be found in addition to 3ds Max ones.

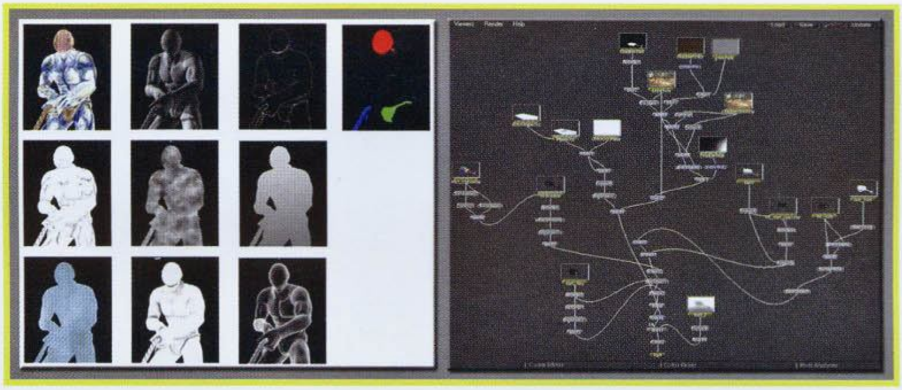


RENDER ELEMENTS

VRay Render Elements

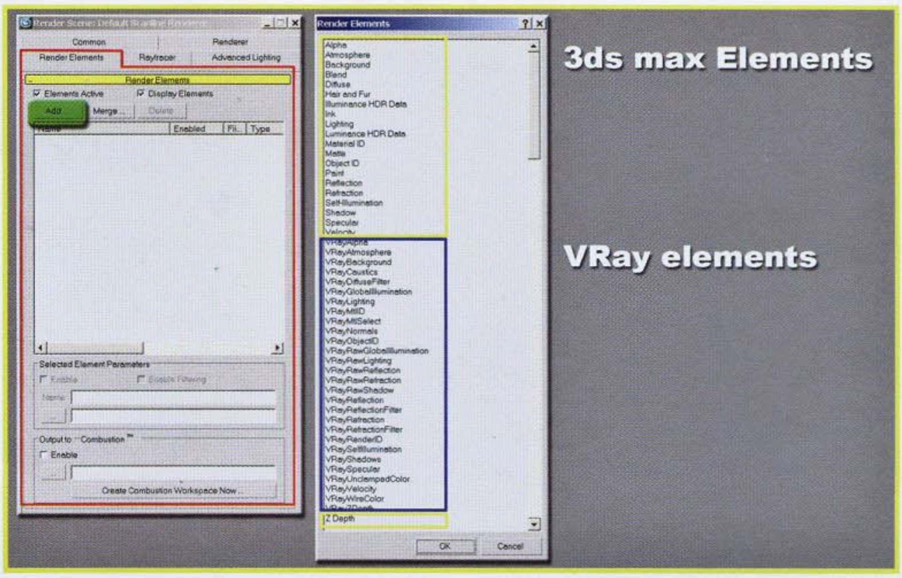
INTRODUCTION

The ***Render Elements*** system is an integrated modulus of 3ds Max, allowing one to separate the information present in a render into single images. ***VRay*** just exploits some of its features. A rendering is composed of objects, meshes, shadows, reflections, refractions, etc. With the ***Render Elements*** it is possible to create a separate render for each one of the listed effects. With appropriate softwares it is possible to use these layers for various purposes, like building the final render by using the separated renders.



■ **Figure 8.1**
On the left some Render Pass. On the right the compositing of the various levels carried out in post-production.

The main purpose of the separation of a render into elements is giving the possibility of modifying its properties in post-production. For example, there is no need to recompute a render if shadows are too dark. One just has to modify the "Shadows" layers in post-production with a software like Adobe After Effects or Autodesk Combustion in order to obtain the changes needed, saving in such a way a lot of time.
The ***Render Elements*** panel is found in the Render scene tool, in the Render Elements card. By clicking on the Add button it is possible to insert the desired layers. Notice how specific ***VRay*** channels are to be found in addition to 3ds Max ones.



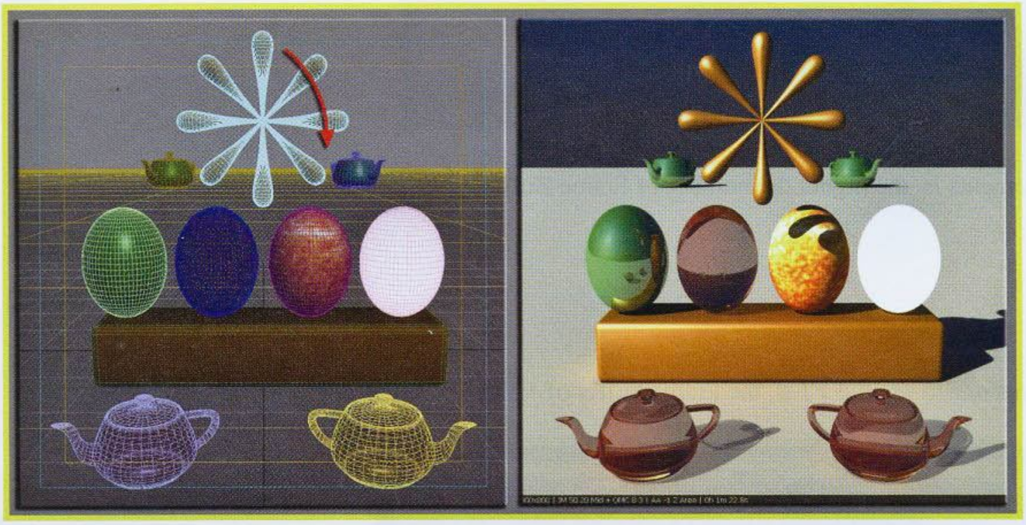
■ **Figure 8.2**
The number of Passes available in VRay is definitely higher compared to 3ds Max. Furthermore, the 3ds Max **Render Elements** are not compatible with VRay's. Vice versa, VRay's ones are not compatible with 3ds Max's.

PARAMETERS

The ***Render Elements*** available in *VRay* are total 28 in the v1.5RC3 release.

■ **Figure 8.3**
28 elements in VRay.

Each one of these elements has unique features, thanks to which it is possible to isolate a certain property, like shadows, reflections, Global Illumination or a color.
In the next tests a scene composed of simple geometries, to which some *VRay* shaders are applied, has been used. Some examples of shaders are chromed shaders, transparent shaders, etc… The whole of this is lit by a ***Sun+Sky*** system. Furthermore, in the centre there is an animated geometry, with a rotating movement. Everything is computed with the ***IM+QMC*** method, ***AA Adaptive subdivs*** -1 2, ***Linear Color mapping***. In the Preference Settings of 3ds Max gamma correction is deactivated. Everything is linear.

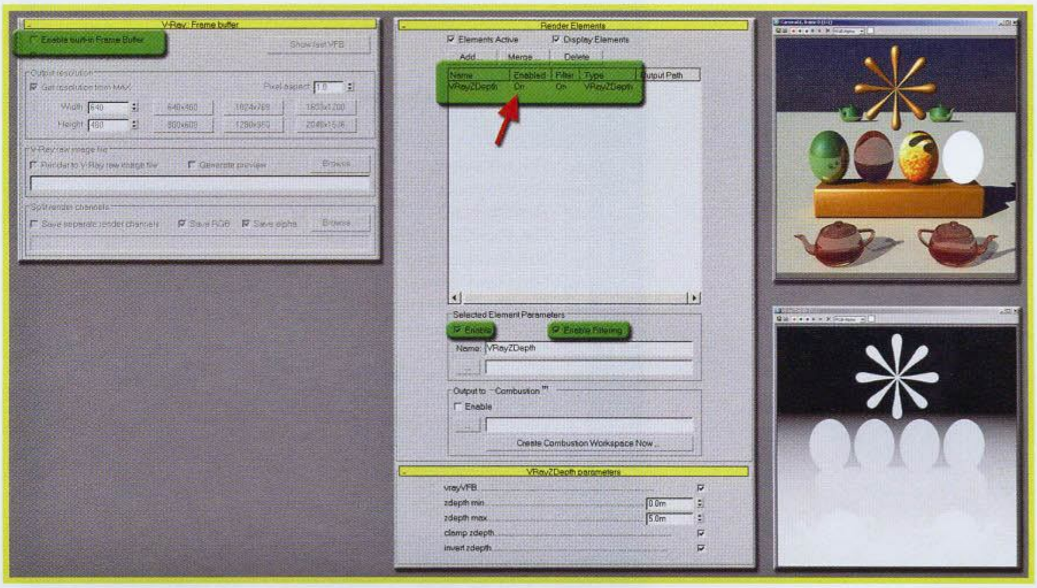


■ **Figure 8.4**
The scene has many elements among which reflections, refractions, colors, textures, shadows, depth, chromed/transparent objects and Global Illumination.

The problem is understanding how to visualize and save the different channels. Three possible solutions exist, by using the Frame Buffer of 3ds Max, the ***VFB*** or the proprietary image format ***.vrimg***.

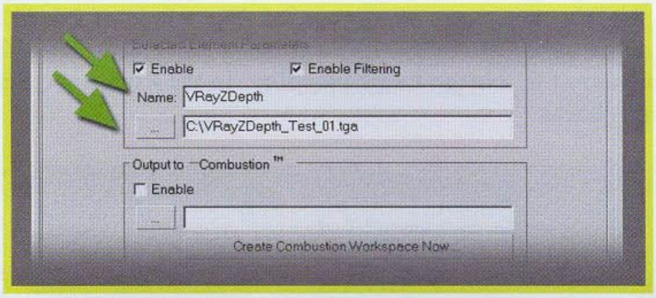
1st Method

For understanding these three systems, the channel ***VRayZDepth*** is going to be employed, for illustrating purposes. It is a layer allowing one to establish the distance of an object from the point of view in order with the coloration of the pixels with a grey scale. First of all the ***VRayZDepth*** is added to the list. Filtering is also activated. The filter allows the application of ***antialiasing*** to the considered object. The ***VFB*** is switched off for the moment.



■ **Figure 8.5**
The rendering of the ***VRayZDepth*** channel inside the FrameBuffer of 3ds Max. Moreover the parameters in the second rollout ***VRayZDepth parameters***, dedicated only to the ***VRayZDepth*** element, are explained. Notice the option ***Enable*** is active and how it is correctly shown in the table with the ***On*** text.

Once the render is complete, inside 3ds Max another Frame Buffer appears in addition to its rendering Frame Buffer, where the chosen channel is shown. Now two possibilities are present: manually saving the channel by clicking on the "floppy disk" symbol shown in the Frame Buffer, or activating automatic saving, by showing the saving name and the path. These options are always available in the ***Render Elements*** panel.

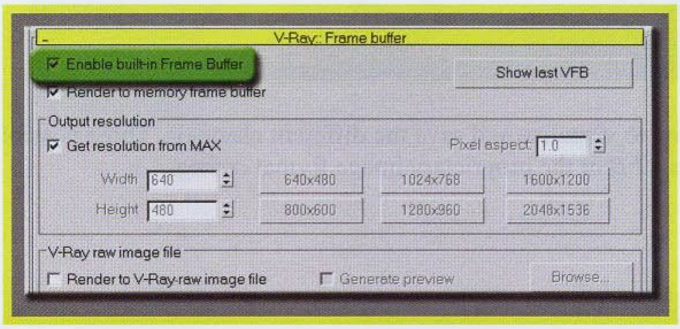


■ **Figure 8.6**
Automatic saving of the ***VRayZDepth*** element.

The higher the number of elements is, the higher the number of Frame Buffer generated inside 3ds Max viewport.

2nd Method

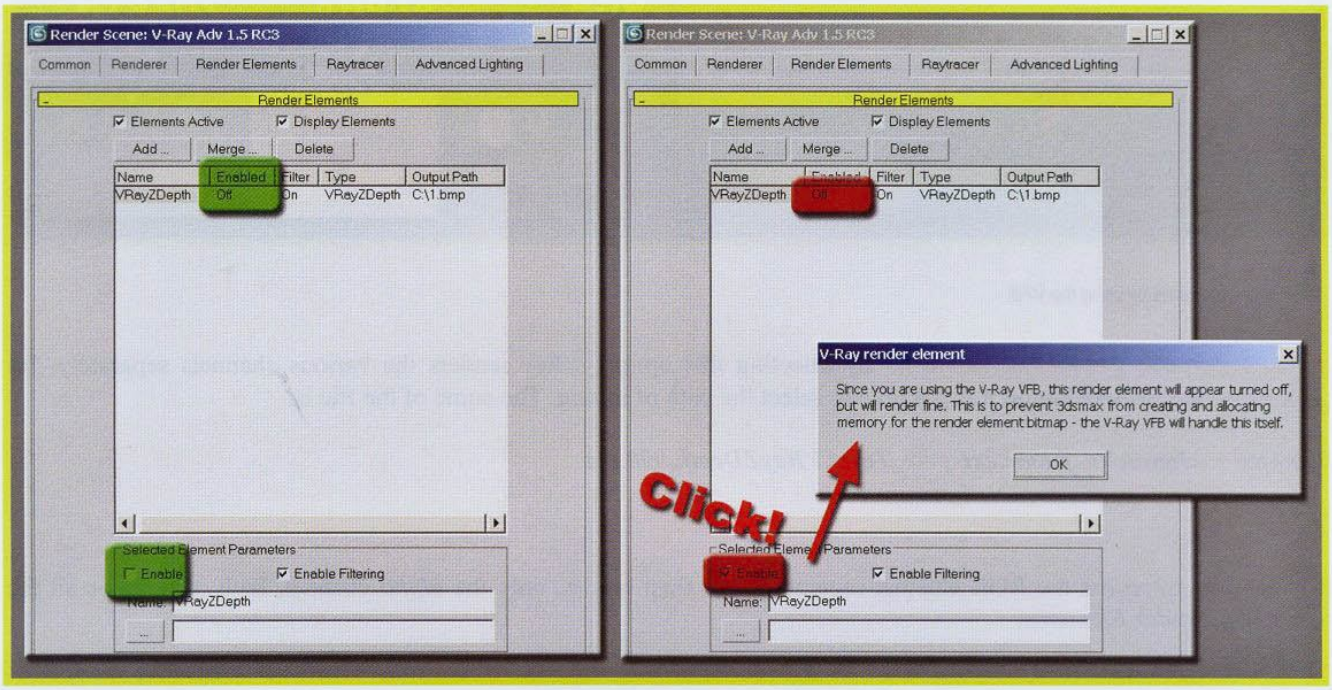
The second procedure requires one to use the ***VFB***, the Frame buffer of ***VRay***.



■ **Figure 8.7**
Activation of the ***VFB***.

Once activated, the Enable option in the ***Render Elements*** panel is turned off. If one tries to reactivate it, ***VRay*** generates a message:

"Since you are using the VRAY VFB, this render element will appear turned off, but will render fine. This to prevent 3dsmax from creating and allocating memory for the render element bitmap - the VRay VFB will handle itself."



■ **Figure 8.8**
The advertising message VRay generates.

VRay generates the various elements directly inside its ***VFB***, without creating any new ones, as in the case of the Frame Buffer of 3dsMax. To see them it is enough to use the pull-down menu located in the ***VFB*** on the upper left. The ***alpha*** channel is always present.

■ **Figure 8.9**
Inside the ***VFB*** all the elements can be manually saved.

It is possible to save the channels both manually, by clicking on the "floppy disk" symbol present in the ***VFB***, or automatically. This time the previous method is no longer valid. To save it one must use the parameters available in the rollout ***VRay: Frame buffer***.

■ **Figure 8.10**
Saving the channels by using the ***VFB***.

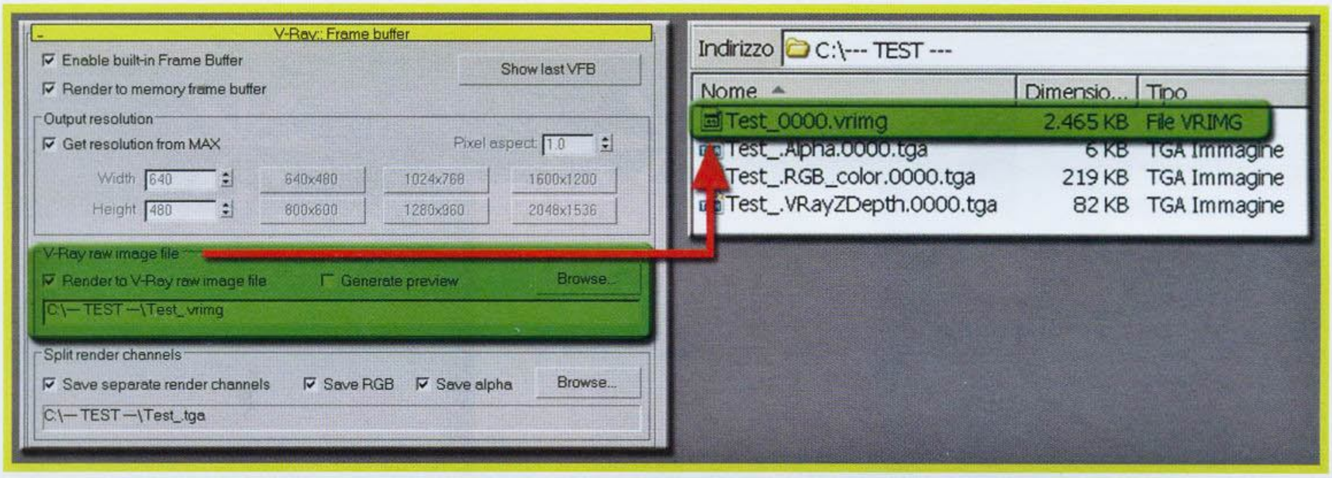
Save separate render channel - by selecting this option, *VRay* renders the various channels separately. By clicking on the ***Browse*** button it is possible to select the path of saving. The name of the file is:

filename + ***.channel*** + ***.frame.ext*** => ***Test_.VRayZDepth.000.tga***

It is possible to render the RGB channel separately, the final render, only the ***alpha*** channel, both of them or all the remaining ***Render Elements***.

3rd Method

The method consists of using the proprietary format of *VRay* ***.vrimage***. This format allows one to save all the channels inserted in the ***Render Elements*** panel inside it. This way just one, big ,file containing all the channels is generated.

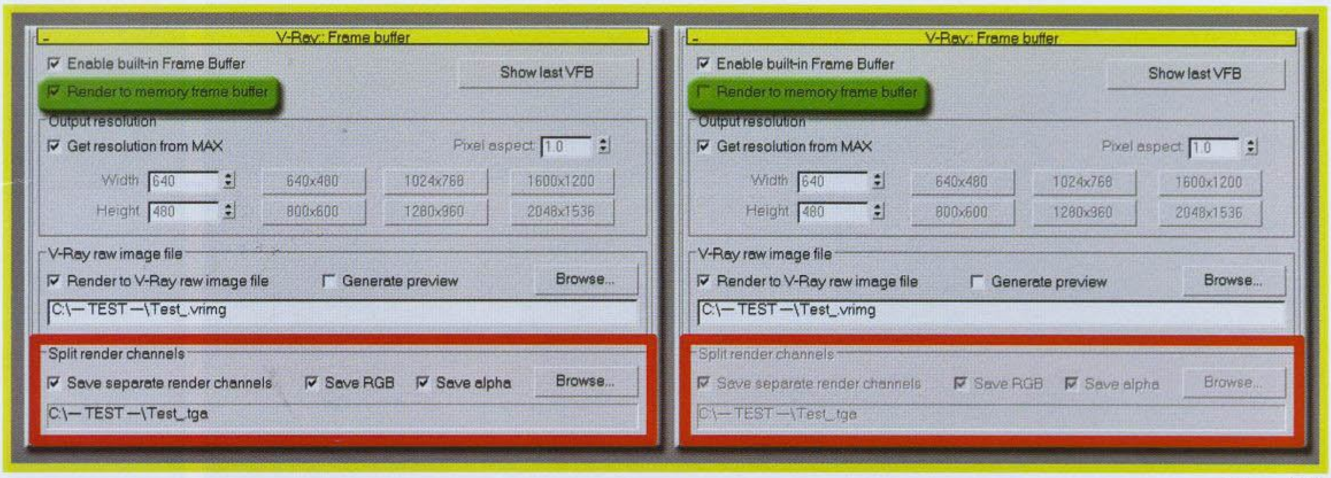


■ **Figure 8.11**
Saving the channels by using the ***.vrimg*** format.

Render to VRay raw image file - by selecting this option, *VRay* renders the channels in one file with extension ***.vrimg***. By clicking on the ***Browse*** button it is possible to select the path of saving. The name of the file will be:

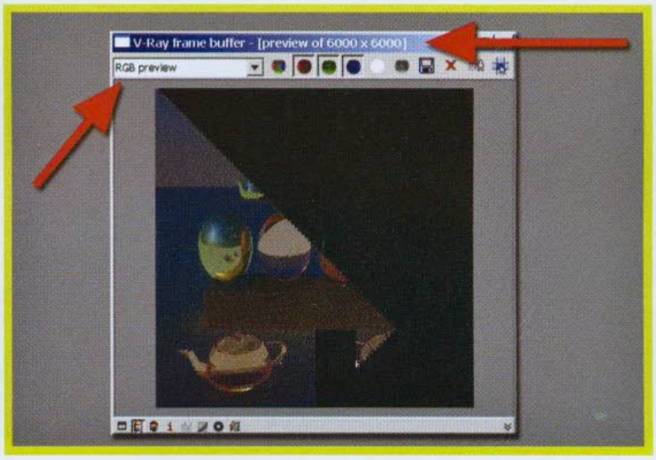
filemane + ***n°frame.ext*** => ***Test_0000.vrimg***

Generate preview - rendering files at a high resolution requires a vast amount of memory from the computer. The higher the resolution is, the more RAM is required, until, above a certain resolution, 3ds could crash. The resources are used by the ***Frame buffer***, both that in 3ds Max and that of *VRay*. For this reason, in the rollout ***VRay: Frame buffer***, the option ***Render to memory frame buffer*** has been introduced. By deactivating the ***VFB***, as one can imagine, *VRay* does not show the user any render, therefore saving memory. As a consequence, also the channels are not available. The ***Split render channels*** section, in fact, is deactivated.



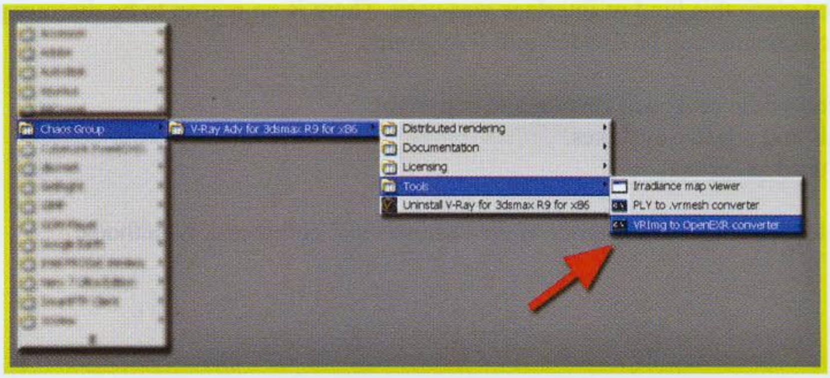
■ **Figure 8.12**
Deactivation of the ***Render to memory frame buffer***.

The section dedicated to the generation of the ***.vrimg*** file remains active in any case. Even though it is no longer possible to observe what *VRay* is computing through the ***VRay frame buffer***, the result can be saved in a format containing all the ***Render Elements***.
The ***Generate preview*** option allows the creation of a very low resolution preview compared to the final resolution. In such a way it is possible to observe what *VRay* is computing. For example, it is possible to stop *VRay* in case of mistakes, have feedback in real time, and save a lot of RAM.



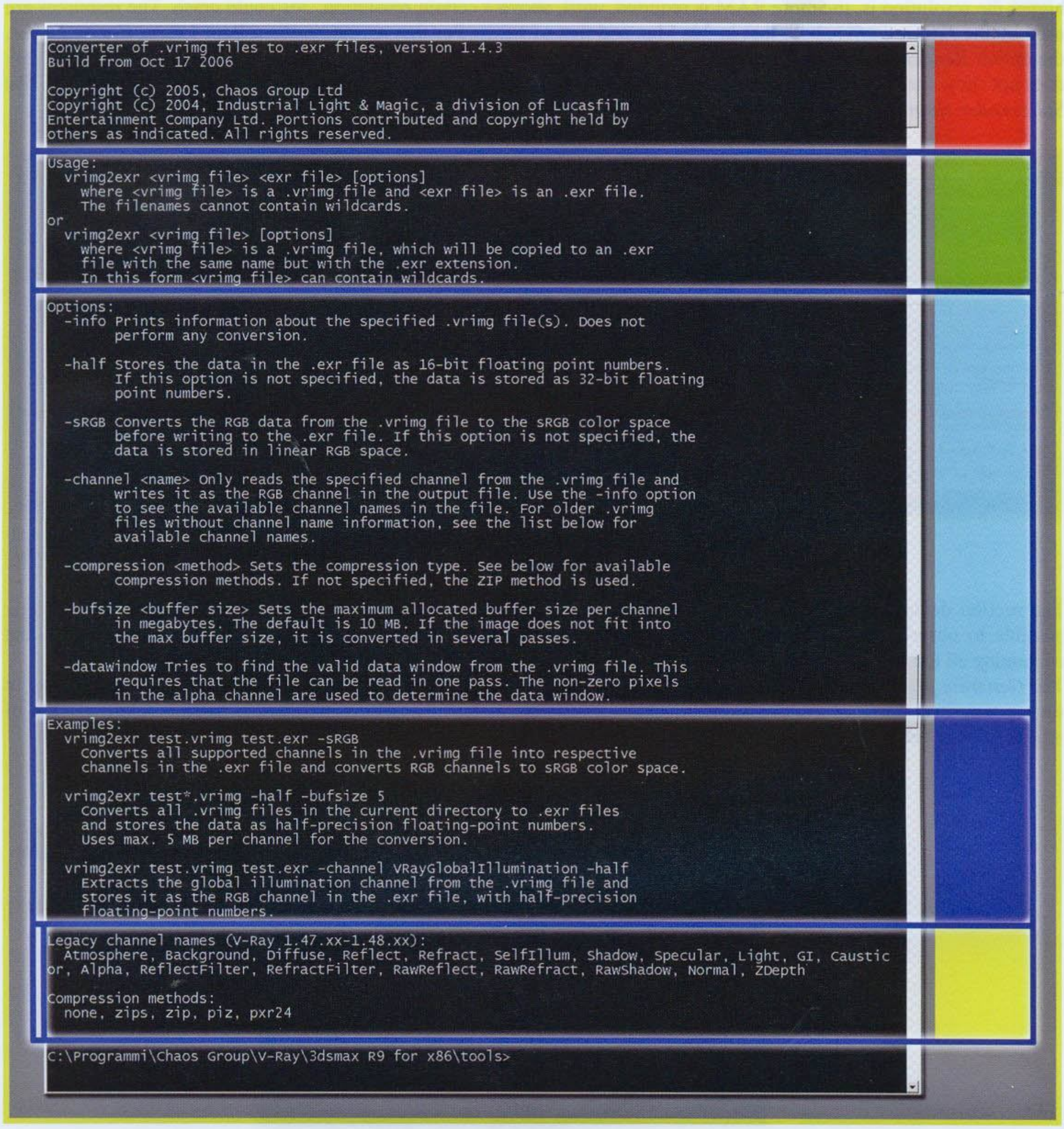
■ **Figure 8.13**
6000 x 6000 pixels resolution render. Notice the word "***preview***" both on the channel and in the title of the window as well.

The last step is to extract the various layers from the ***.vrimg*** file. For this purpose, ***Chaosgroup*** has prepared a little utility: ***VRimg to OpenEXR***.



■ **Figure 8.14**
The program for extracting the channels from the ***.vrimg*** file is located inside the folder for VRay installation.

The DOS display for managing all the commands of the conversion program.

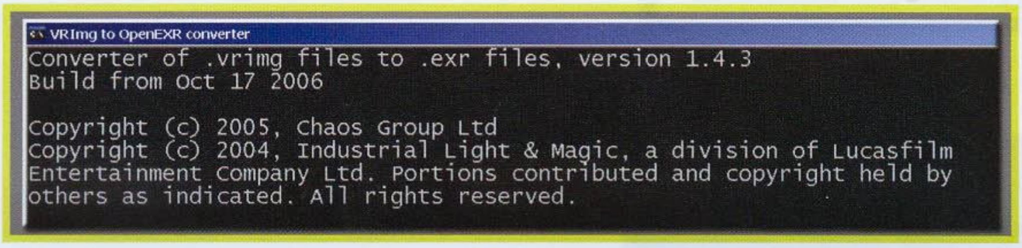


■ **Figure 8.15**
The command window for the conversion of the ***.vrimg*** file.

Wow! A long time has passed since this kind of interface was used. There is no reason to be discouraged, as it is easier than one might expect. The commands can be divided into five groups.

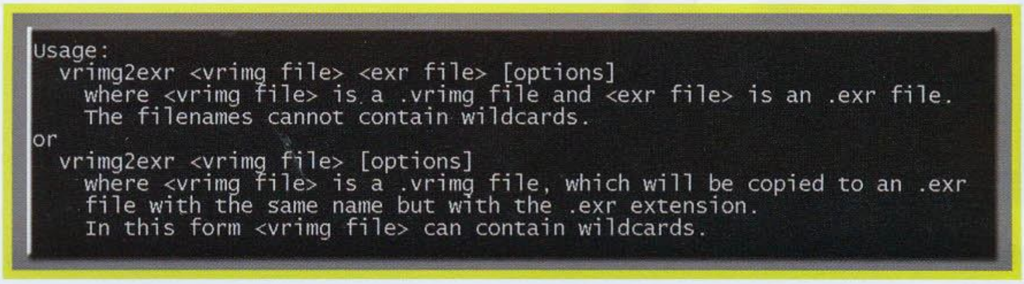
Group 1 - generic information of the program, version and copyright.
Group 2 - the real commands and relative syntaxes.
Group 3 - the options to be used as syntax.
Group 4 - some examples.
Group 5 - other parameters to use, such as the names of the channels and compression methods.

The general information is contained in the first group: a brief description of its purpose, the version, the creation date and the Copyright in ***.vrimg*** format of ***Chaosgroup*** ownership and the .exr format of IL&M ownership.



■ **Figure 8.16**
General information of the program.

The second group, perhaps the most important one, shows the exact syntax to be used when the extracting relative channels from the ***.vrimg*** file.



■ **Figure 8.17**
The syntax for program execution.

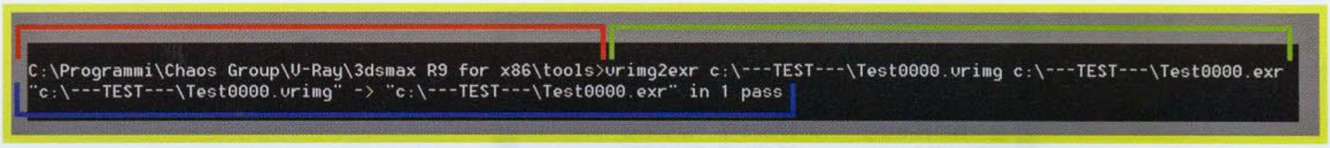
Once the program has started, a DOS display is present, with a flashing cursor. The folder is the *VRay* installation folder.

■ **Figure 8.18**
The folder where the program is contained.

In this directory, \tools, the program is present. First of all the program must be recalled. There are two ways of doing this. The first one means one has to write in the DOS window:

vrimg2exr ***<vrimg_file>*** ***<exr_file>*** [options]

vrimg2exr	= the program is activated.
<vrimg file>	= the name of the .vrimg file must be inserted, like, for example, test0000.vrimg.
<exr file>	= the name of the destination .exr file must be inserted like, for example, test0000.exr.
[options]	= additional options, not necessary.



■ **Figure 8.19**
An example of a green command.

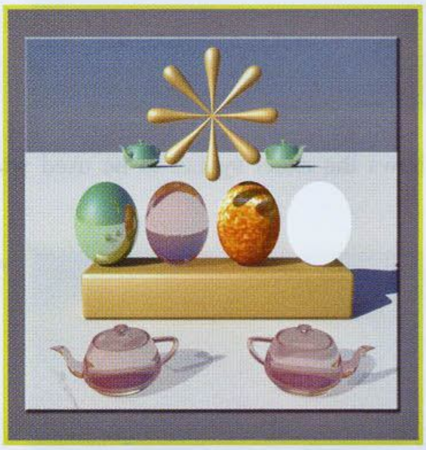
In this example, the file Test0000.virmg generated by *VRay* has been saved in the folder C:\---TEST---. For this reason the command highlighted by the green bracket contains both the input file ***.vrimg*** path and the output one in .exr format. *VRay* answers by showing what it is doing, that is, the analysis of the ***.vrimg*** file and the exportation in .exr format, as shown in the blue bracket.

■ **Figure 8.20**
The original ***.vrimg*** file and the ***.exr*** file obtained by the conversion.

Nome	Dimensione	Tipo
Test0000.exr	8.241 KB	File EXR
Test0000.vrimg	17.521 KB	File VRIMG

Observe with Photoshop the .exr file obtained with the conversion.

■ **Figure 8.21**
Visualization of the ***.exr*** file obtained by the conversion.

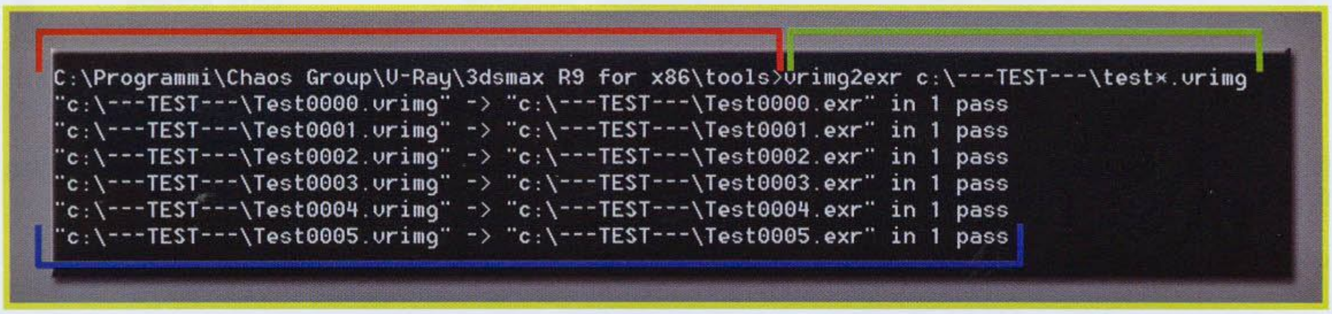


The file is correct, but something is wrong in the "luminosity" (in truth it concerns the gamma). The image is too bright. Now we will see how to correct it.

The second method concerns the conversion of multiple ***.vrimg*** files, generated in an animation. In this case progressive alpha-numerical files are available. For example: Test0000.vrimg, Test0001.vrimg, Test0002.vrimg, Test0003.vrimg, Test0004.vrimg, Test0005.vrimg, etc…In this case the syntax to be used is the following:

vrimg2exr <***vrimg_wildcard***> [options]

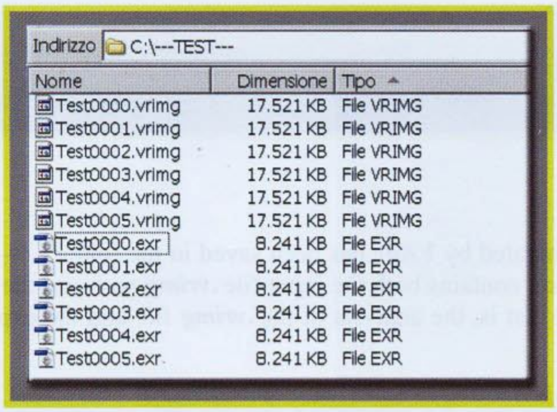
<***vrimg_wildcard***> = name of the file containing the DOS symbols such as * or ?. In this case the files are automatically converted in .exr format. In both methods the original ***.vrimg*** files are not deleted.



■ **Figure 8.22**
Multiple conversion of the ***.vrimg*** file.

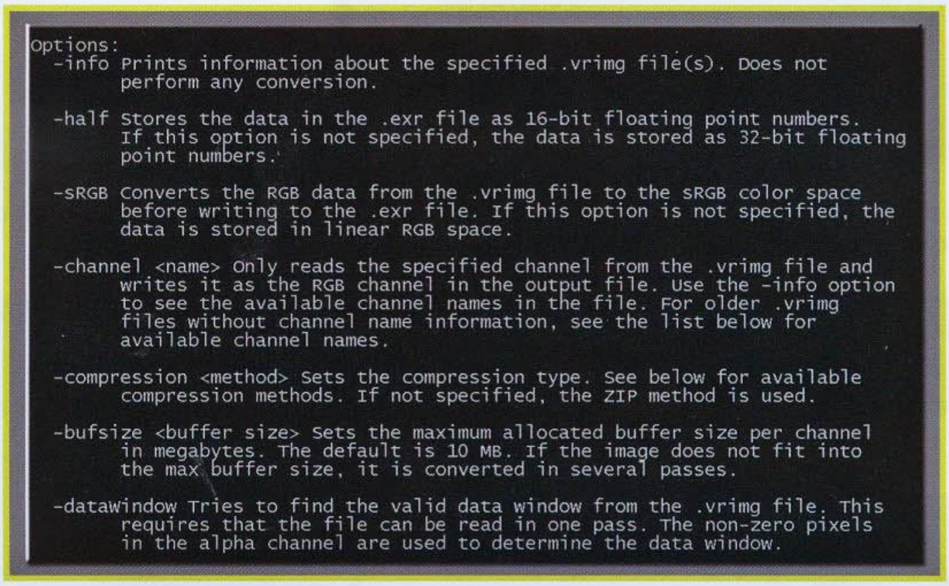
The conversion of five ***.vrimg*** files returns the automatic generation of as many .exr files.

■ **Figure 8.23**
The ***.vrimg*** files and the ***.exr*** files obtained by a multiple conversion.



Nome	Dimensione	Tipo
Test0000.vrimg	17.521 KB	File VRIMG
Test0001.vrimg	17.521 KB	File VRIMG
Test0002.vrimg	17.521 KB	File VRIMG
Test0003.vrimg	17.521 KB	File VRIMG
Test0004.vrimg	17.521 KB	File VRIMG
Test0005.vrimg	17.521 KB	File VRIMG
Test0000.exr	8.241 KB	File EXR
Test0001.exr	8.241 KB	File EXR
Test0002.exr	8.241 KB	File EXR
Test0003.exr	8.241 KB	File EXR
Test0004.exr	8.241 KB	File EXR
Test0005.exr	8.241 KB	File EXR

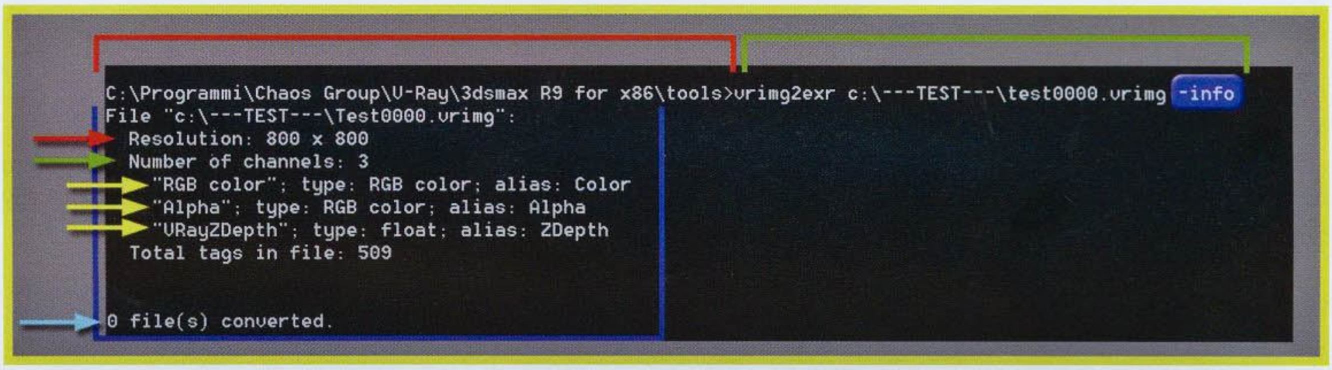
The third group contains the options and the relative explanations. By observing the syntax in Group 1, you might remember the presence of the [options]. It is possible to insert, instead of that word, one of the seven commands available in the ***vrimg2exr.exe*** program.



```
Options:
  -info Prints information about the specified .vrimg file(s). Does not
        perform any conversion.

  -half Stores the data in the .exr file as 16-bit floating point numbers.
        If this option is not specified, the data is stored as 32-bit floating
        point numbers.

  -sRGB Converts the RGB data from the .vrimg file to the sRGB color space
        before writing to the .exr file. If this option is not specified, the
        data is stored in linear RGB space.

  -channel <name> Only reads the specified channel from the .vrimg file and
        writes it as the RGB channel in the output file. Use the -info option
        to see the available channel names in the file. For older .vrimg
        files without channel name information, see the list below for
        available channel names.

  -compression <method> Sets the compression type. See below for available
        compression methods. If not specified, the ZIP method is used.

  -bufsize <buffer size> Sets the maximum allocated buffer size per channel
        in megabytes. The default is 10 MB. If the image does not fit into
        the max buffer size, it is converted in several passes.

  -dataWindow Tries to find the valid data window from the .vrimg file. This
        requires that the file can be read in one pass. The non-zero pixels
        in the alpha channel are used to determine the data window.
```

■ Figure 8.24
The options to be used in the ***vrimg2exr.exe*** program.

-info - this parameter allows one to obtain information on the analyzed ***.vrimg*** file. Its use does not generate any conversion.



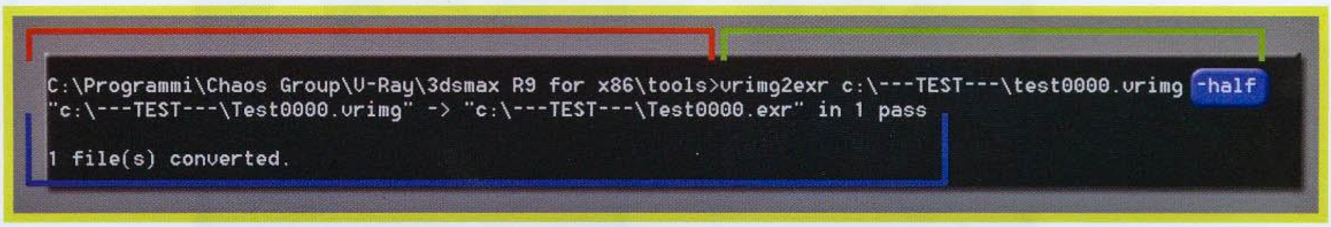
```
C:\Programmi\Chaos Group\V-Ray\3dsmax R9 for x86\tools>vrimg2exr c:\---TEST---\test0000.vrimg -info
File "c:\---TEST---\Test0000.vrimg":
  Resolution: 800 x 800
  Number of channels: 3
    "RGB color"; type: RGB color; alias: Color
    "Alpha"; type: RGB color; alias: Alpha
    "VRayZDepth"; type: float; alias: ZDepth
  Total tags in file: 509

0 file(s) converted.
```

■ Figure 8.25
Employment of the ***-info*** option.

For using it, it is enough to write as the last word: ***-info***. Instantly ***VRay*** gives back the information on the ***.vrimg*** file, in accordance with the size of the file and the number of contained channels. Proceeding in order:

Resolution	800x800
Number of channels......	3
Name of the channels contained ...	RGB Color, Alpha, ***VRay***ZDepth
Number of converted files	0

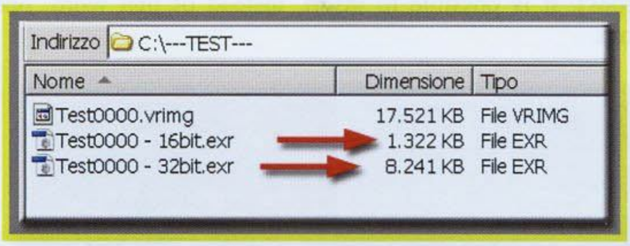
-half - by using this parameter, the converter is going to create a ***16bit*** floating point .exr file. If not expressly specified, the ***vrimg2exr.exe*** program is going to generate ***32bit*** floating point images always. The difference, other than the size in Bytes, relies in the precision, which is greater when a ***32bit*** format file is used.



```
C:\Programmi\Chaos Group\V-Ray\3dsmax R9 for x86\tools>vrimg2exr c:\---TEST---\test0000.vrimg -half
"c:\---TEST---\Test0000.vrimg" -> "c:\---TEST---\Test0000.exr" in 1 pass

1 file(s) converted.
```

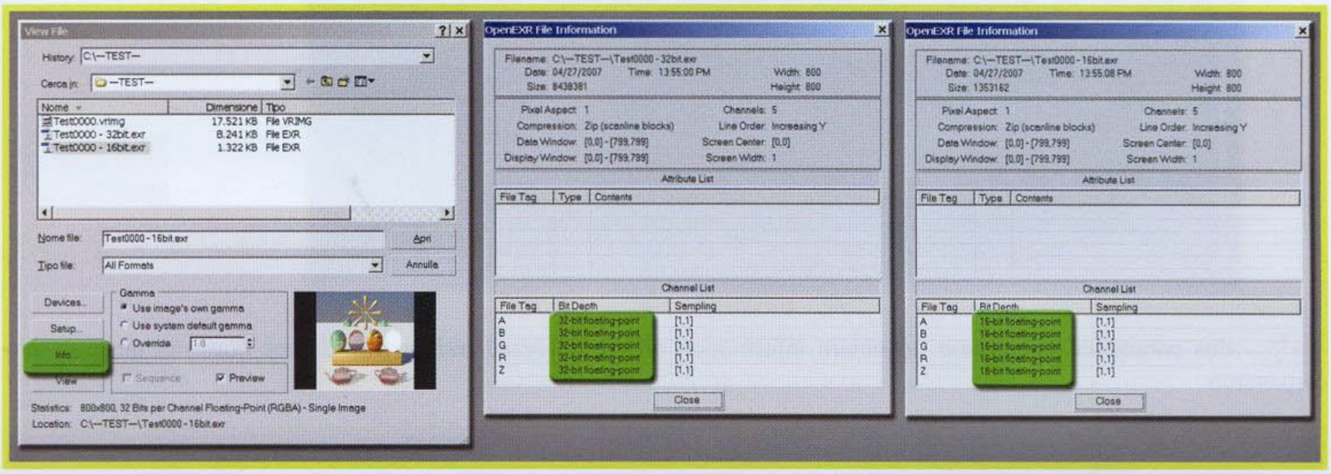
■ Figure 8.26
–half option employment.

By observing the file one can understand how much the variation between ***16bit*** and ***32bit float number*** affects its size.



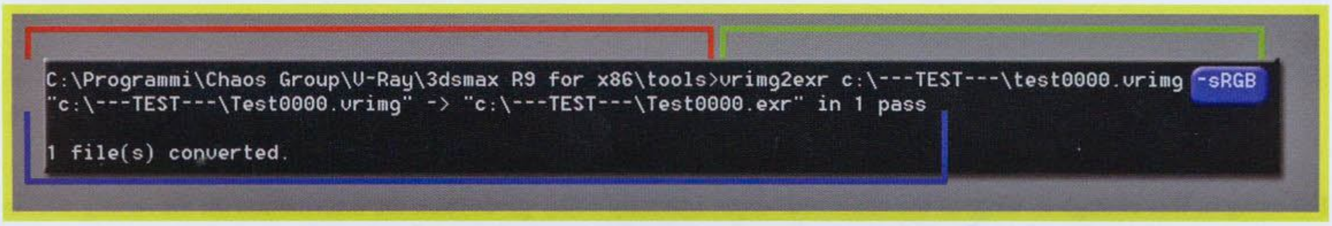
■ **Figure 8.27**
Difference between a ***16bit*** file and a ***32bit*** in terms of Bytes.

If the files are opened with the ***viewer*** of 3ds Max through the command ***File -> View Image file*** and the ***Info*** option s selected, it is possible to establish the real conversion to ***16bit***.



■ **Figure 8.28**
With the Info option located in the Viewer of 3ds Max it is possible to obtain information on the ***.exr*** files.

-sRGB - this parameter converts the channel inside the ***.vrimg*** file from the ***RGB*** color space to the ***sRGB*** one before writing the file in the .exr format. In the case of skipping the ***.sRGB*** option, the conversion is not going to apply any kind of correction on the color.

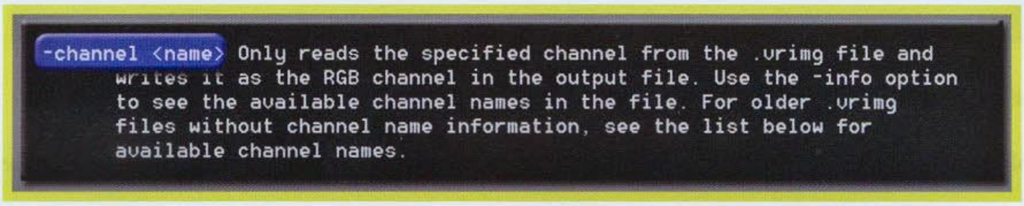


■ **Figure 8.29**
Use of the ***-sRGB*** option.

Compare the two .exr files obtained with and without the ***-sRGB*** option.

■ **Figure 8.30**
Files with and without the ***sRGB*** correction.

-channel - by using this option it is possible to extract a specific channel from the ***.vrimg*** file. To do this, as well as using the ***-channel*** command, it is necessary to let the program know which channel one wants to extract. No channel is going to be generated, except the indicated one.



```
-channel <name> Only reads the specified channel from the .vrimg file and
      writes it as the RGB channel in the output file. Use the -info option
      to see the available channel names in the file. For older .vrimg
      files without channel name information, see the list below for
      available channel names.
```

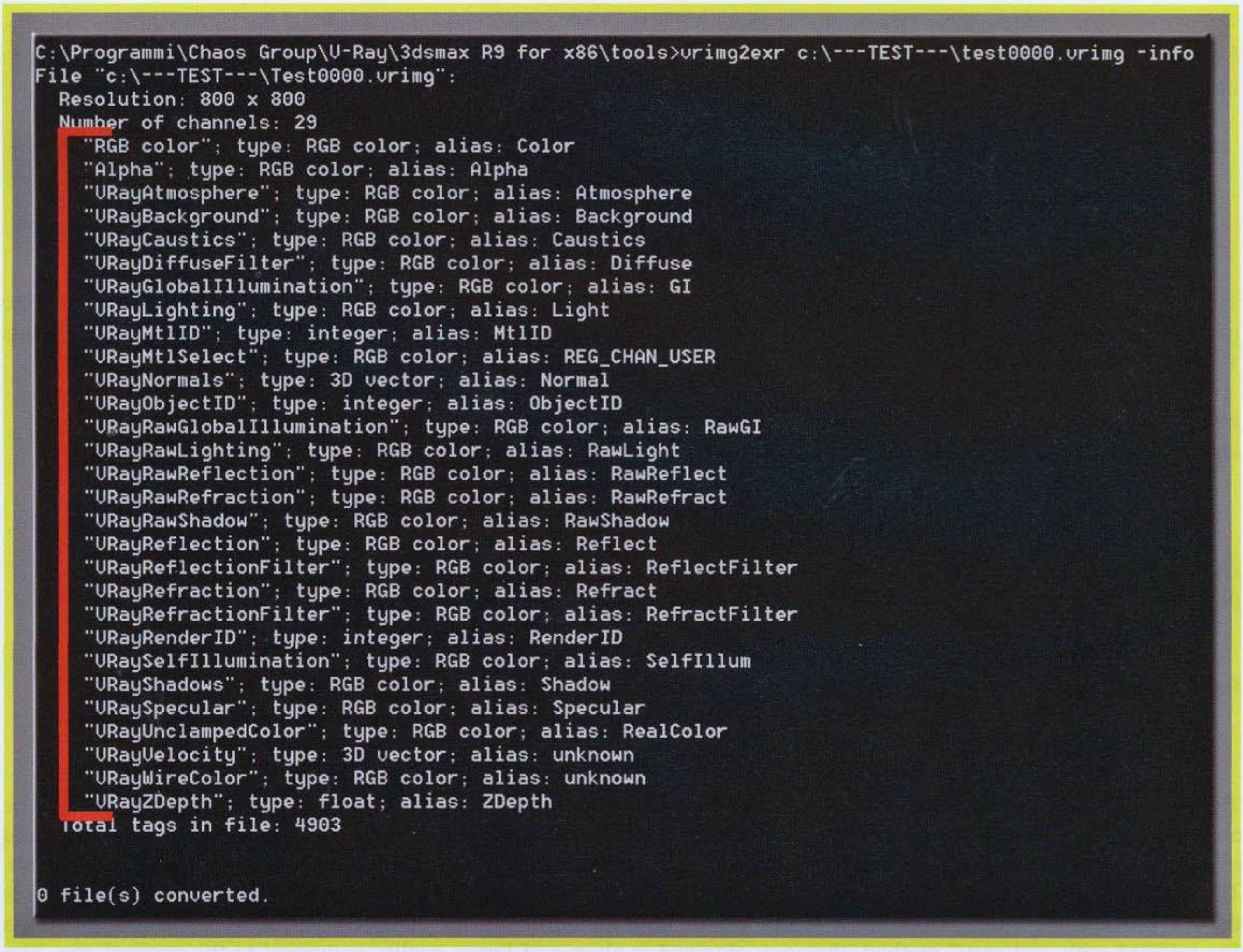
■ **Figure 8.31**
The syntax to be used with the ***-channel*** option.

The correct syntax, as is suggested by the same program., is:

-channel <*name*>

<*name*> = name of the channel one wants to extract.

The channels supported until today are: ***Atmosphere***, ***Background***, ***Diffuse***, ***Reflect***, ***Refract***, ***SelfIllum***, ***Shadow***, ***Specular***, ***Light***, ***GI***, ***Caustics***, ***RawGI***, ***RawLight***, ***RealColor***, ***RGB Color Alpha***. The channels ***VRayMtlID***, ***VRayObjectID***, ***VRayRenderID*** and ***VRayZDepth***, although present in the ***.vrimg*** file, cannot be extracted yet from the current ***vrimg2exr.exe*** program. Their exportation has not yet been implemented by the programmers. In order to know which channels are really present in the ***.vrimg*** file one just has to use the ***-info*** command. In this case the ***.vrimg*** file contains 3 channels.
In the next image the ***.vrimg*** file of the same scene is shown, with all the channels within it.



```
C:\Programmi\Chaos Group\V-Ray\3dsmax R9 for x86\tools>vrimg2exr c:\---TEST---\test0000.vrimg -info
File "c:\---TEST---\Test0000.vrimg":
  Resolution: 800 x 800
  Number of channels: 29
    "RGB color"; type: RGB color; alias: Color
    "Alpha"; type: RGB color; alias: Alpha
    "VRayAtmosphere"; type: RGB color; alias: Atmosphere
    "VRayBackground"; type: RGB color; alias: Background
    "VRayCaustics"; type: RGB color; alias: Caustics
    "VRayDiffuseFilter"; type: RGB color; alias: Diffuse
    "VRayGlobalIllumination"; type: RGB color; alias: GI
    "VRayLighting"; type: RGB color; alias: Light
    "VRayMtlID"; type: integer; alias: MtlID
    "VRayMtlSelect"; type: RGB color; alias: REG_CHAN_USER
    "VRayNormals"; type: 3D vector; alias: Normal
    "VRayObjectID"; type: integer; alias: ObjectID
    "VRayRawGlobalIllumination"; type: RGB color; alias: RawGI
    "VRayRawLighting"; type: RGB color; alias: RawLight
    "VRayRawReflection"; type: RGB color; alias: RawReflect
    "VRayRawRefraction"; type: RGB color; alias: RawRefract
    "VRayRawShadow"; type: RGB color; alias: RawShadow
    "VRayReflection"; type: RGB color; alias: Reflect
    "VRayReflectionFilter"; type: RGB color; alias: ReflectFilter
    "VRayRefraction"; type: RGB color; alias: Refract
    "VRayRefractionFilter"; type: RGB color; alias: RefractFilter
    "VRayRenderID"; type: integer; alias: RenderID
    "VRaySelfIllumination"; type: RGB color; alias: SelfIllum
    "VRayShadows"; type: RGB color; alias: Shadow
    "VRaySpecular"; type: RGB color; alias: Specular
    "VRayUnclampedColor"; type: RGB color; alias: RealColor
    "VRayVelocity"; type: 3D vector; alias: unknown
    "VRayWireColor"; type: RGB color; alias: unknown
    "VRayZDepth"; type: float; alias: ZDepth
  Total tags in file: 4903


0 file(s) converted.
```

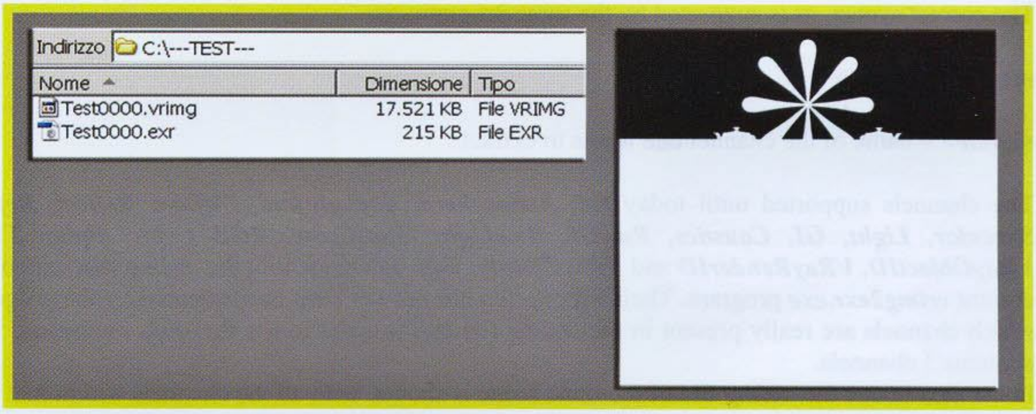
■ **Figure 8.32**
All the available channels. There are 28 in the image and not 29, because the "***RGB Color***" channel, relative to the complete rendering, is also included.

The value <*name*> corresponds to the name of the channel, like ***VRayAtmosphere*** etc…If ***.vrimg*** files previous to the 1.5 version are used (v 1.47.xx v1.48.xx) the names of the channels correspond to those shown in the Group 5.

■ **Figure 8.33**
Use of the ***-channel*** option. One clarification: the ***RGB Color*** must be written between the inverted commas "***RGB Color***".

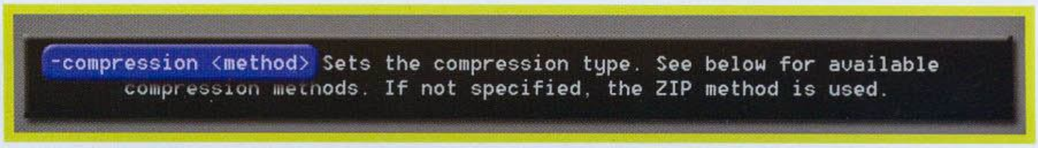
This way it has been possible to create a ***32bit*** floating point file with the same name of the ***.vrimg*** file, containing the alpha channel only.

■ **Figure 8.34**
The ***alpha*** channel extracted by the ***.vrimg*** file.



-compression - this option allows one to choose the kind of compression to use during the saving of the .exr file. Similarly to the ***-channel*** option, in this case "options of options" exist, allowing one to decide which compression to use. The available ones are: none, ***zip, zips, piz, pxr24***.

■ **Figure 8.35**
The ***-compression*** option must always be followed by the compression method.



The .exr file, which is an ***HDRI*** format, fills a lot of space since it contains a lot of information. For this reason the necessity of using compression algorithms arose in order to reduce the space and increase the efficiency. The formats ***VRay*** uses are a total of five.

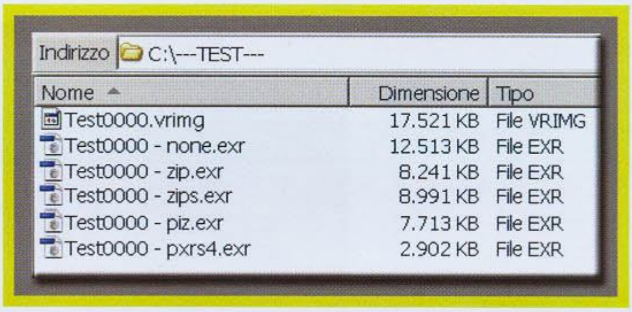
none Use this option when you do not want any kind of compression.
zip Method compressing the image by blocks of 16 lines (scanline). The horizontal differences between adjacent pixels are compressed by using the Open source zlib libraries. The decompression of the file is faster than the ***piz*** method, but the compression is slower. It compresses between 45%-55%. Lossless.
zips Equal to the previous one, but is compressed one line at a time. Lossless.
piz This system uses a new combined Wavelet/Huffman method of compression. Files are compressed and decompressed more or less at the same speed. It is very efficient, most of all when very grainy and disturbed images are involved, reducing the file around 35% -55%. Lossless.
Pxr24 Compression developed by Loren Carpenter in collaboration with Pixar. After the reduction of the image from 32bit float to 24bit float by means of a rounding off, the horizontal difference between adjacent pixels is compressed with zlib libraries. Lossy.

The ***vrimg2exr.exe*** program allows one to compress images. To open them again, the visualization program must be able to read the compressed data, interpret them and rebuild the original images. The lossless compression is that kind of compression which does not lead to a loss of information during the data compression-decompression phase. Therefore, the images are always the same, both before and after the compression. Vice versa, the Lossy compression is the compression algorithm which leads to the loss of information during the compression phase. By decompressing a file fulfilled with a "Lossy" method, the copy obtained is always of lower quality compared to the original one. The amount of this loss of quality depends on the compression settings of the program.

The syntax is very similar to the previous one:

■ **Figure 8.36**
Use of the ***-compression*** option.

Here below, the compressed files:

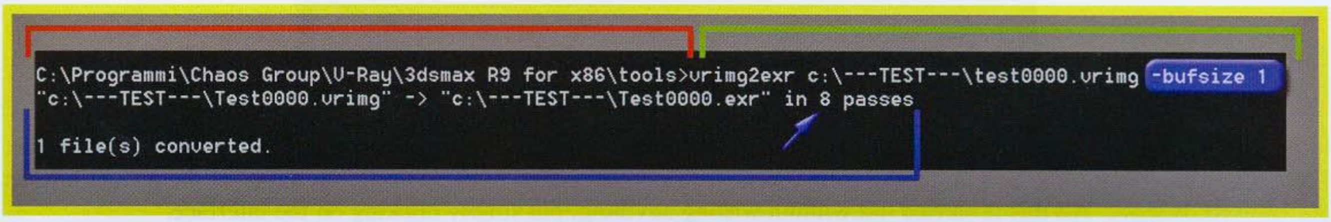


Nome	Dimensione	Tipo
Test0000.vrimg	17.521 KB	File VRIMG
Test0000 - none.exr	12.513 KB	File EXR
Test0000 - zip.exr	8.241 KB	File EXR
Test0000 - zips.exr	8.991 KB	File EXR
Test0000 - piz.exr	7.713 KB	File EXR
Test0000 - pxrs4.exr	2.902 KB	File EXR

■ **Figure 8.37**
.exr files obtained with different compression options.

The smallest file has been generated with the Lossy compression ***pxr24***, reduced almost three times compared to the other methods. For Lossless formats the best one is the ***piz*** method.

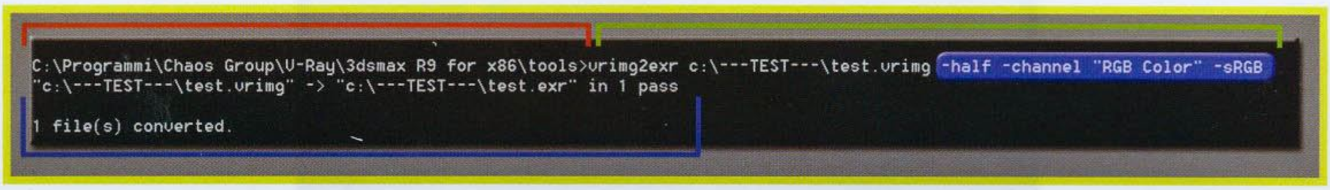
-bufsize - this parameter allows one to set the maximum size of the ***buffer*** used for the conversion from ***.vrimg*** to .exr. If the image is bigger, the process is to be carried out in several steps. A major ***buffer*** means a more rapid conversion, but a higher request of memory as well. The default value is set to 10 MB for each channel. In this case as well an "option inside the option" is present, consisting in the indication of the memory to be devoted to the conversion process.



■ **Figure 8.38**
-bufsize option syntax.

By observing the previous images, the size of the .exr file extracted from the ***.vrimg*** file is roughly 8MB. The passages of the conversion are a total of eight if a 1MB buffer is used.

The options we have seen so far can be inserted one after the other so as to obtain a great number of combinations.



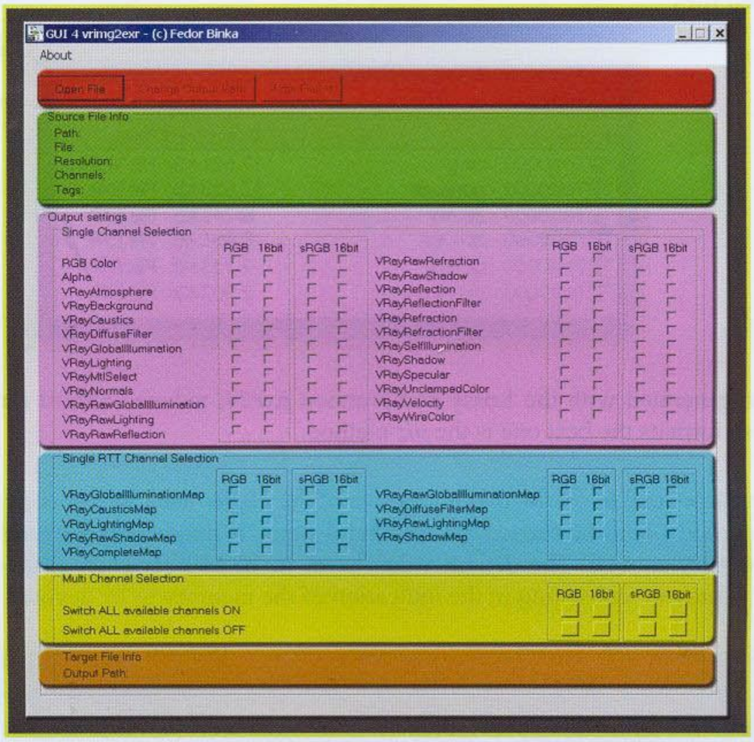
■ **Figure 8.39**
Example of multiple syntax.

In this case a ***16bit*** file .exr is extracted from the ***.vrimg*** file with the same name. The extracted channel is to be the ***RGB color*** in the ***sRGB*** color space.

The fourth group has some examples with relative comments. Each word can be understood with a bit of attention.

The fifth group shows the names of the channels to be used in case one wants to extract the pass renders from a ***.vrimg*** file, previous to the version v1.5 of ***VRay***. Lastly, always in the fifth group, the compression methods, already discussed, are listed.

After this interesting illustration of the employment of the conversion program ***vrimg2exr.exe*** by means of a command line, and while waiting for an interface by ***Chaosgroup***, it is owing to thank a user, since he has anticipated the software house itself by accomplishing an interface for the ***vrimg2exr.exe*** program. It is a little program written specifically for ***VRay*** by the author Fedor Binka (**http://binka.biz/?Download**). Once started, a very compact interface appears, divided into six zones:



■ **Figure 8.40**
The interface of the ***GUI 4 vrimg2exr***. The six zones allow one to:

1. Manage the ***.vrimg*** file.

2. Obtain information on the ***.vrimg*** file in use.

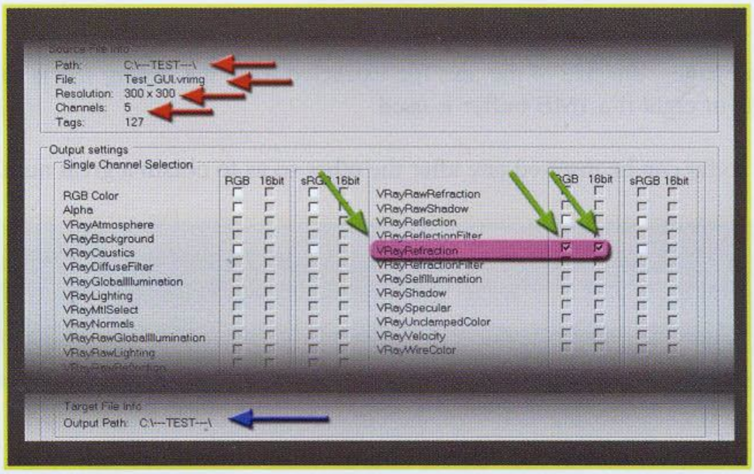
3. Choose the channels to be extracted, with the relative options.

4. Choose the channels to be extracted for the Render to Texture, with the relative options.

5. Select multiple channels.

6. Decide the save path.

In this example, the Test_GUI.vrimg, having a 300x300 pixels resolution, contains 5 channels: "***RGB color***", ***alpha***, ***VRayCaustics***, ***VRayReflection*** and ***VRayRefraction***. It has been chosen to export only the layer of the refractions, in the 16bit RGB space. All this is carried out without writing anything, but simply clicking on the interface.



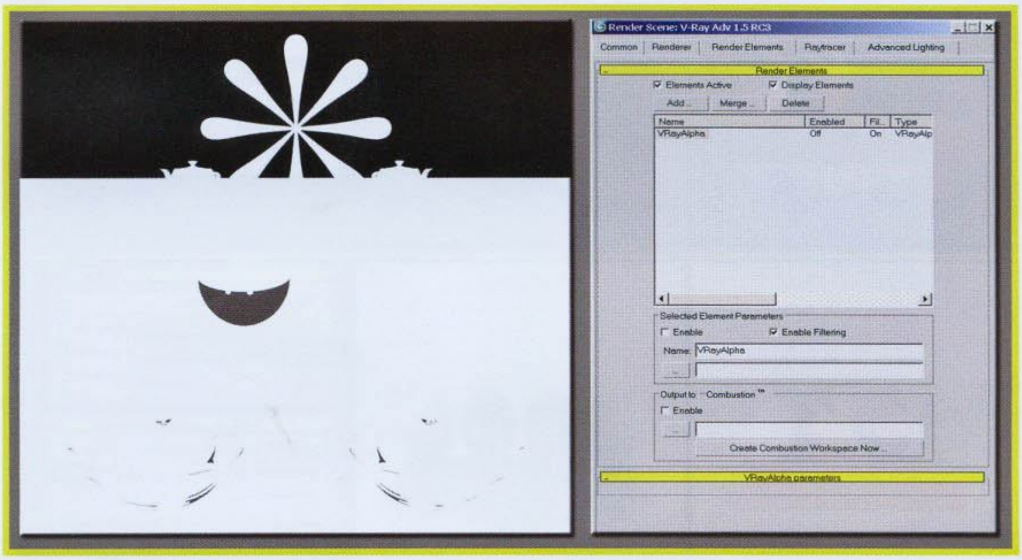
■ **Figure 8.41**
Example of use of the program.

The final result is a .exr format, with the name of the original file, with the suffix of the exported channel added.

■ **Figure 8.42**
The program correctly exports the name of the channel.

The Render Elements

VRayAlpha - this layer allows one to render the ***alpha*** channel.

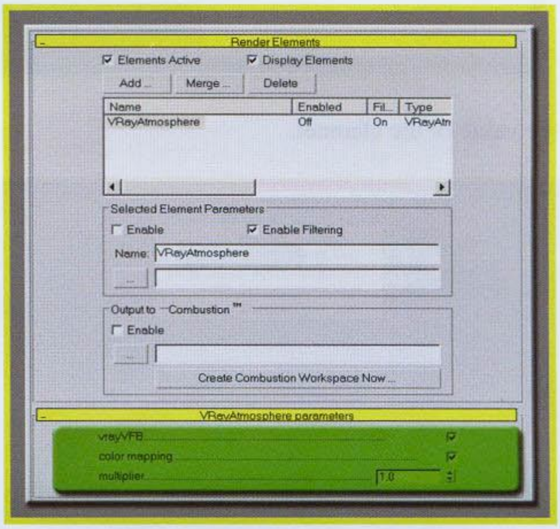


■ **Figure 8.43**
The ***VRayAlpha*** channel. It does not have additional parameters.

The channel supports:

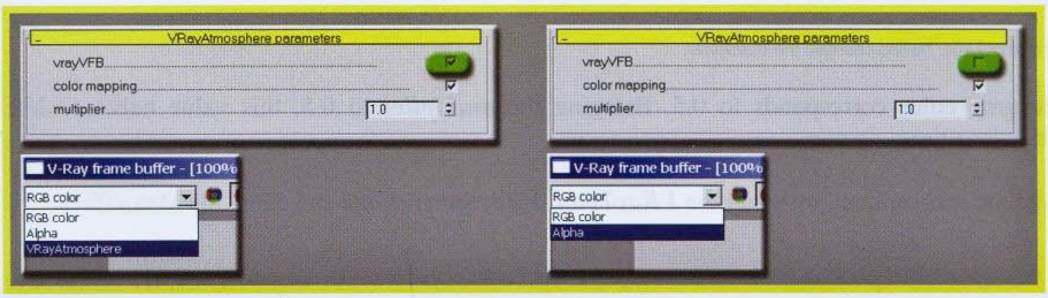
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayAtmosphere - this layer allows one to represent on one image, the atmospheric effects. For example the ***VRayToon***, ***VRaySphereFade***, ***Fog*** etc… in respect to the previous channel, new parameters are present.



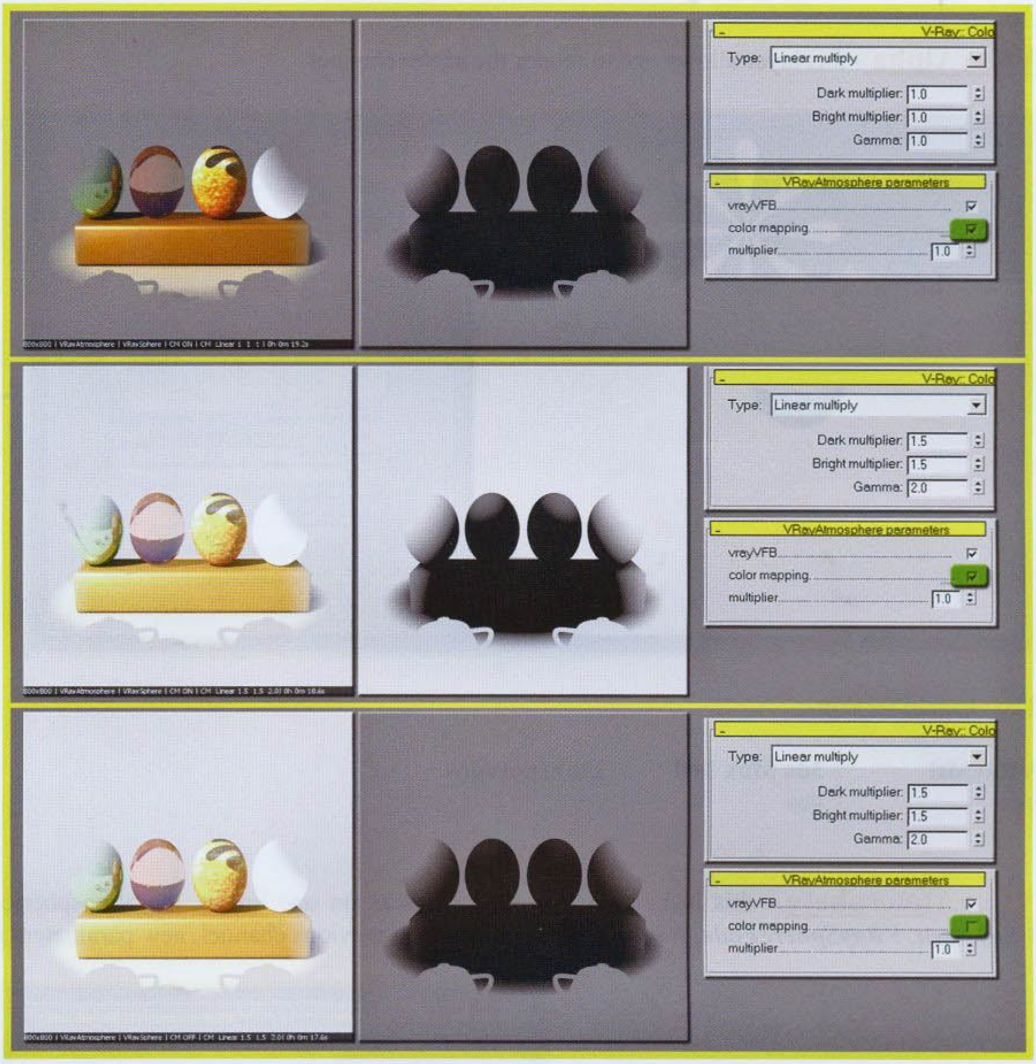
■ **Figure 8.44**
The ***VRayAtmosphere*** channel. Notice the new parameters.

vrayVFB - it enables the generation of the channel within the ***Frame buffer***. Its deactivation does not make the channel appear inside the ***VFB***, nor is it saved as separate channel inside the ***.vrimg*** channel.



■ **Figure 8.45**
The deactivation of the ***vrayVFB*** temporarily switches off the generation of the channel.

color mapping - this function deactivates any setting located in the rollout ***VRay: Color mapping***, bringing the render to linear settings.



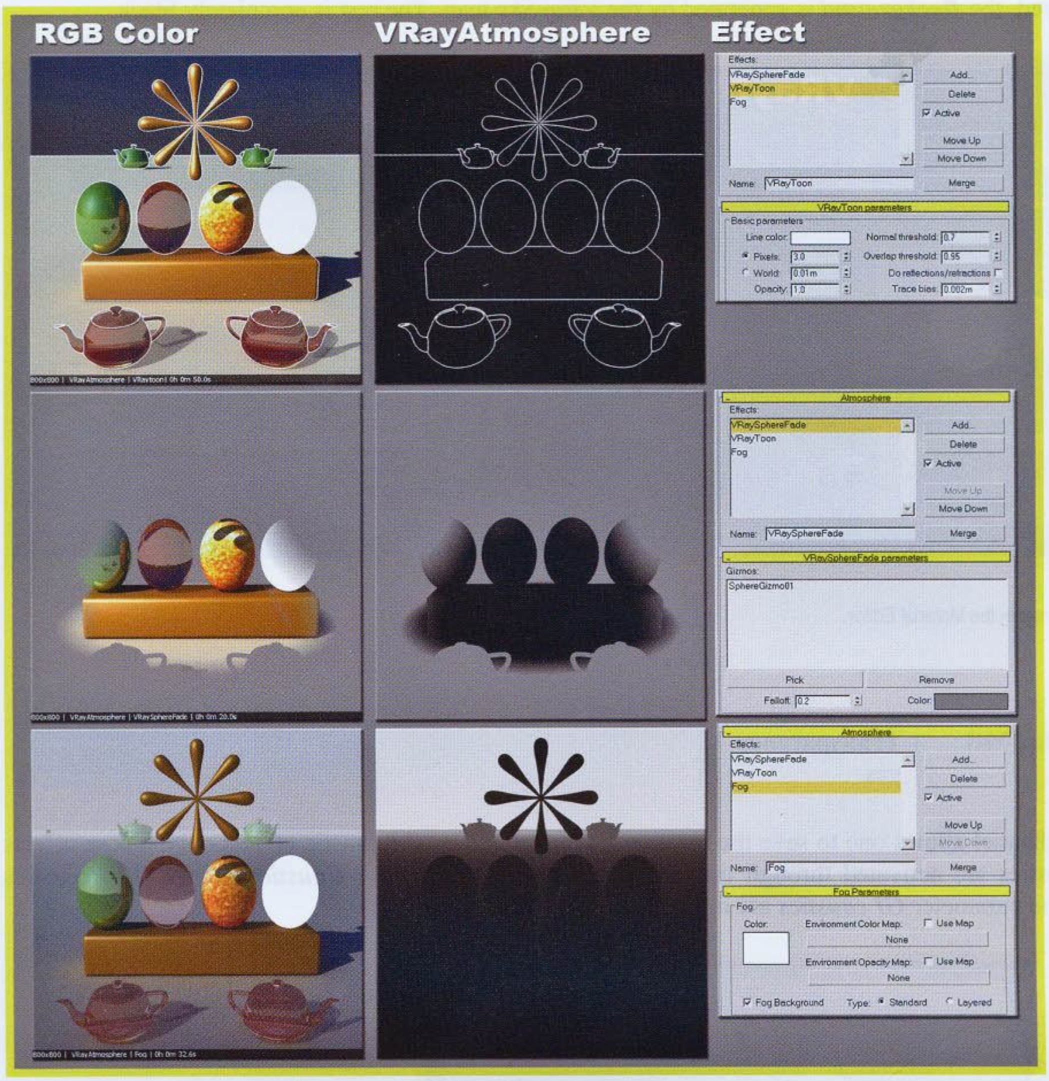
■ **Figure 8.46**
Use of the color mapping parameter: In the first row, both the ***RGB*** and the ***VRayAtmosphere*** channels are rendered with ***Linear Color mapping*** using the ***color mapping*** option of the ***Render elements.*** By modifying the ***Color mapping*** settings the effects are visible on both of the channels, the renders are much brighter. By deactivating the ***color mapping*** option of the VRayAtmosphere channel, ***Color mapping*** no longer has an effect on the render of the atmospheric channel. Instead, only the ***RGB*** channel is effected by it.

Multiplier - it multiplies the value of the channel.

■ **Figure 8.47**
Effects on the ***VRayAtmosphere*** channel after having changed the ***multiplier***.

With ***multiplier*** = 1.0 the grey color corresponds to 0.5. Bringing the ***multiplier*** to 0.5, this value halves, while multiplying it four times, it is four times greater.

Now we just have to analyze the real channel, by using the ***VRayToon***, ***VRaySphereFade*** and ***Fog***, 3ds Max effect.



■ **Figure 8.48**
The ***VRayAtmosphere*** channel.

In the first example the ***VRayToon*** effect has been used. The ***VRayAtmosphere*** generates a colored layer. On the background, completely black, the ***VRayToon*** effect is represented, with the colors set in the same effect. In this case the line is white. If it had been black, it would not have been visible, as ***Background*** would have been of the same color of the outlines.

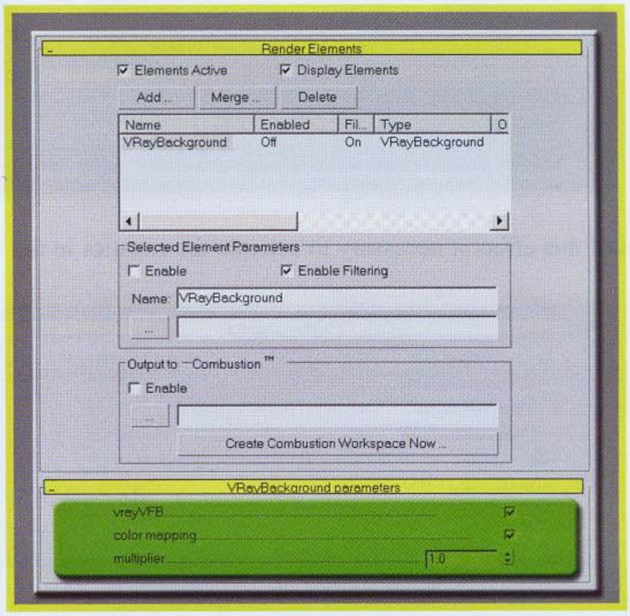
In the second example the ***VRaySphereFade*** effect has been used. The channel is in ***RGB***, as it was in the previous example. Everything within the gizmo is black. Vice versa, the color is the one set inside the same effect.

In the last case the effect is the classic 3ds Max ***Fog***. Also here, the color of the layer, which shows the ***Fog*** effect only, is white, because this is the color set within the ***Atmosphere*** rollout. If the color had been yellow, the ***VRayAtmosphere*** would have been yellow.

This channel supports:

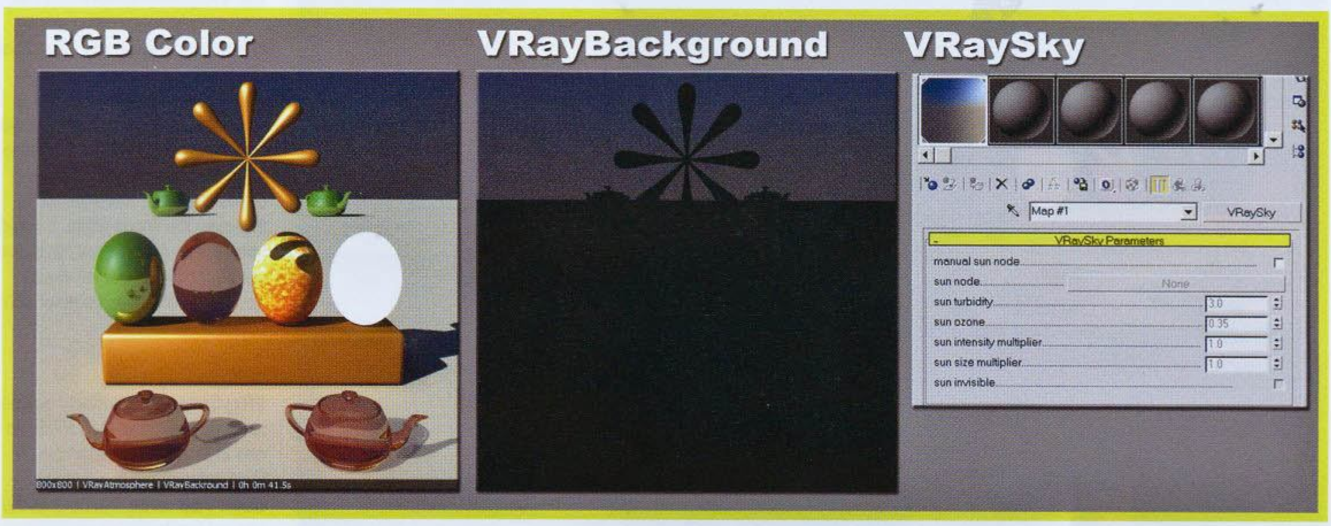
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayBackground - this channel allows one to isolate the background. Everything which is not part of it is black.



■ **Figure 8.49**
The ***VRayBackground*** channel. Notice the new parameters, the same as the ***VRayAtmosphere*** ones.

In the case being considered, the ***Background*** is generated by the ***VRaySky*** map. The rest is completely black.



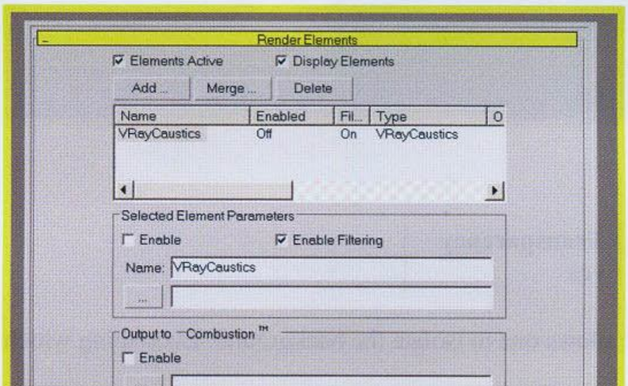
■ **Figure 8.50**
The ***VRayBackground*** and the map in the Material Editor.

The channel supports:

VRayMtl	**3ds Max Mtl**	**Transparency**
yes	yes	yes

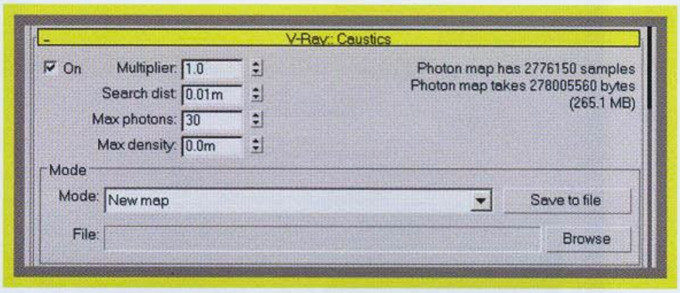
VRayCaustics - this channel allows one to save the information generated by the caustic effects only. These effects are visible when the caustics are activated through the options located in the ***VRay: Caustics*** rollout. The caustics generated via the use of the parameter ***GI caustics*** are not considered.

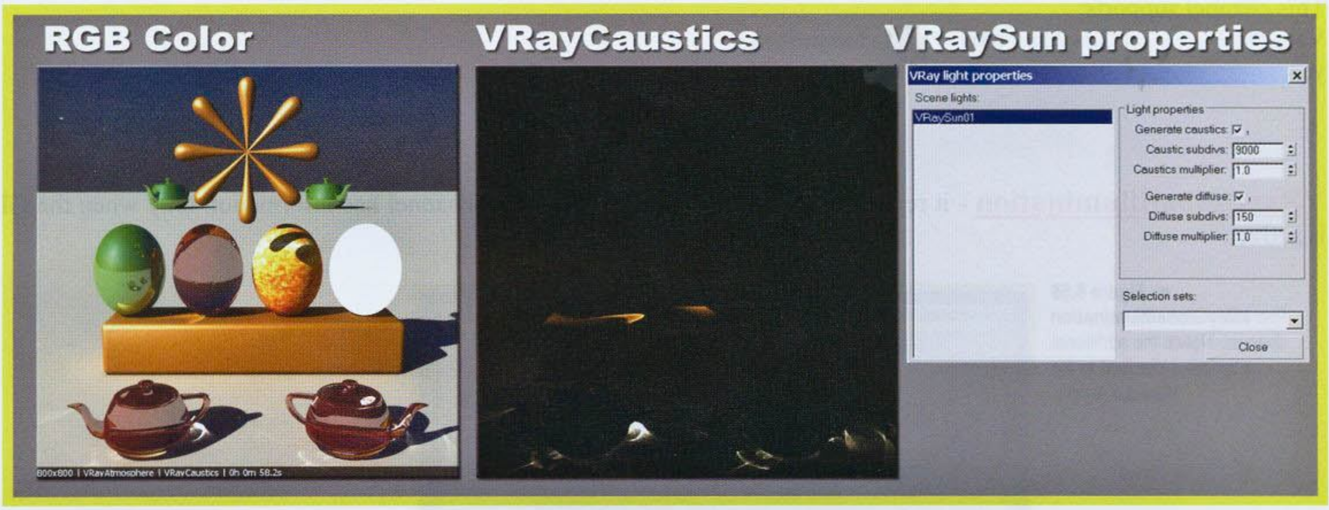
■ **Figure 8.51**
The ***VRayCaustics*** channel. Notice the additional parameters, identical to the ones in the previous Render Pass.



To observe the generation of this effect it necessary to activate the caustics in the ***VRay: Caustics*** rollout.

■ **Figure 8.52**
The activation of the caustic effects and the parameters used.



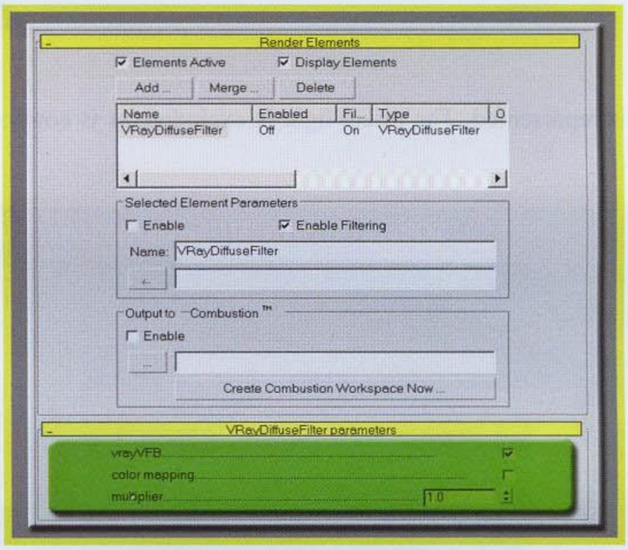


■ **Figure 8.53**
The ***VRayCaustics*** channel.

The channel supports:

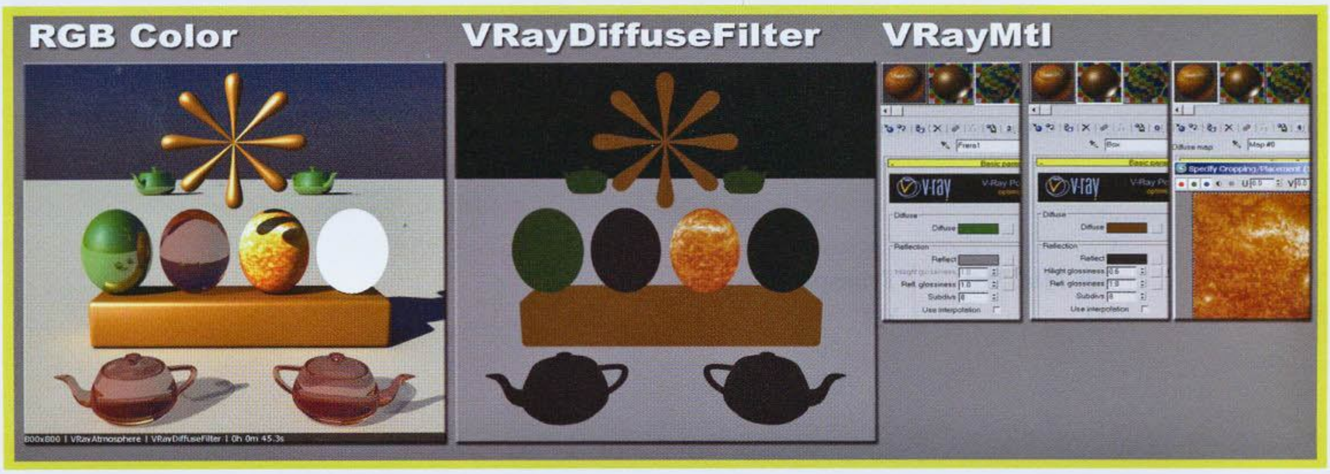
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayDiffuseFilter - this channel represents the ***Diffuse*** part of the shader. It can be represented both by a color or a texture.



■ **Figure 8.54**
The ***VRayDiffuseFilter*** channel. Notice the additional parameters identical to the previous layers.

No color or texture is applied to the ***Background***, since it is not a geometrical object.

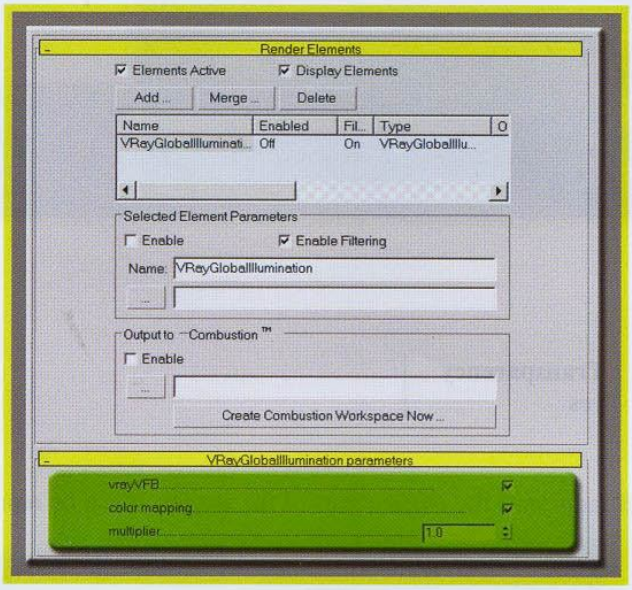


■ **Figure 8.55**
The ***VRayDiffuseFilter*** layer. Notice the complete absence of light.

This channel supports:

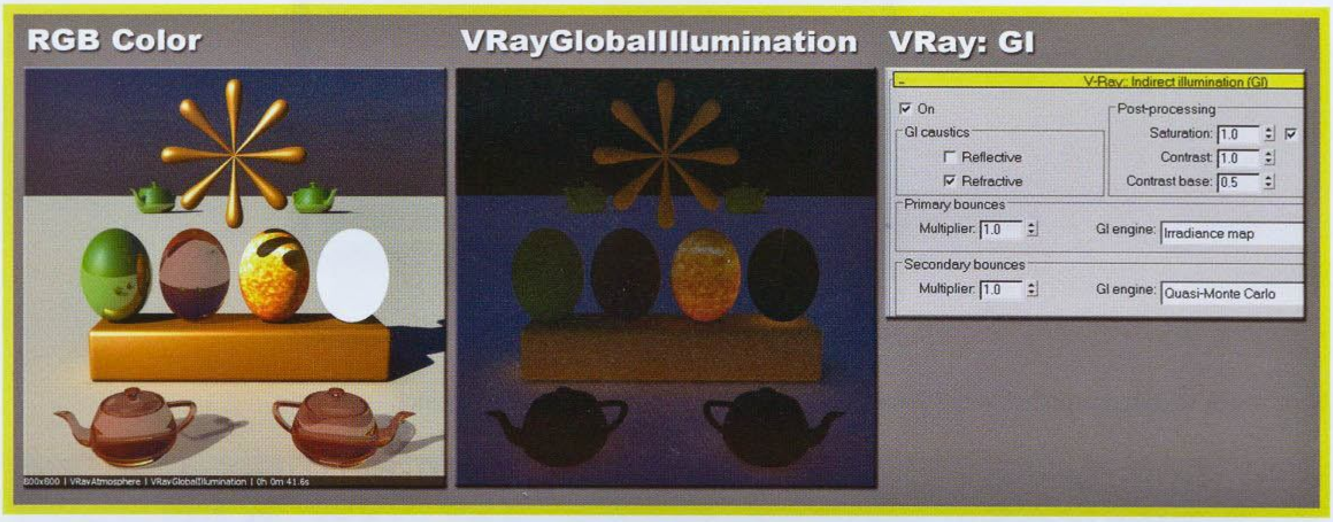
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayGlobalIllumination - it represents the Global Illumination. This channel has information only when the GI is active.



■ Figure 8.56
The *VRayGlobalIllumination* channel. Notice the additional parameters identical to the previous layers.

In this case the GI only is represented. The direct light is excluded, as is confirmed by the absence of direct shadows too.

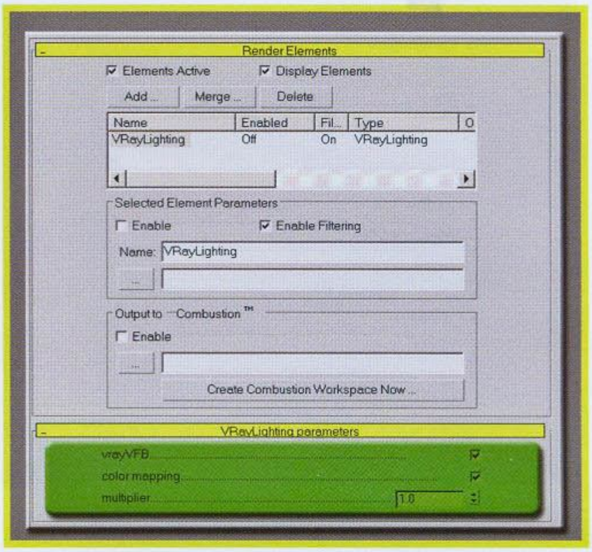


■ Figure 8.57
The *VRayGlobalIllumination* channel and the rendering engines adopted for the computation of the GI.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayLighting - it represents the direct light generated by light sources. There is no presence of GI in this channel.



■ **Figure 8.58**
The ***VRayLighting*** channel. Notice the additional parameters identical to the previous layers.

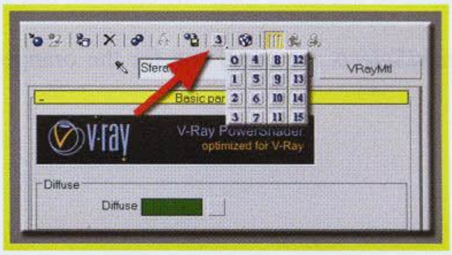
Now the render with the GI active and ***VRayLighting*** channel shall be compared to a render obtained with GI, reflections and refractions deactivated. Notice the similarity between this last render and the ***VRayLighting*** channel. The geometry is completely black with the ***VRayLightingMtl*** in the ***VRayLighting*** channel, since its light is exclusively indirect.

■ **Figure 8.59**
The ***VRayLighting*** channel represents only indirect light.

Questo canale supporta:

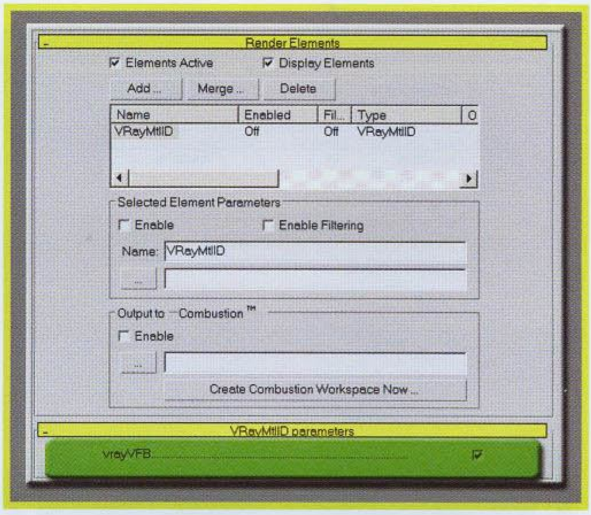
VRayMtl	**3ds Max Mtl**	**Transparency**
yes	yes	yes

VRayMtlID - this channel represents the ID number of the material. This number, by default equal to 0 for every shader, can be found in the Material Editor and is named ***Material ID Channel***. By clicking on the number indicated by the arrow, a table containing 15 numbers appears, which is precisely the ***Material ID Channel***, used by ***VRay***. All the objects this material is assigned to have the same color in the ***VRayMtlID***.



■ **Figure 8.60**
The Material ID Channel.

Unlike the previous channels, the ***VRayMtlID*** Channel has only one extra parameter.



■ **Figure 8.61**
The ***VRayMtlID*** channel. Notice the presence of just one additional parameter.

The ***VRayMtlID*** Channel.

■ **Figure 8.62**
The ***VRayMtlID*** with an enlargement of a portion of the image.

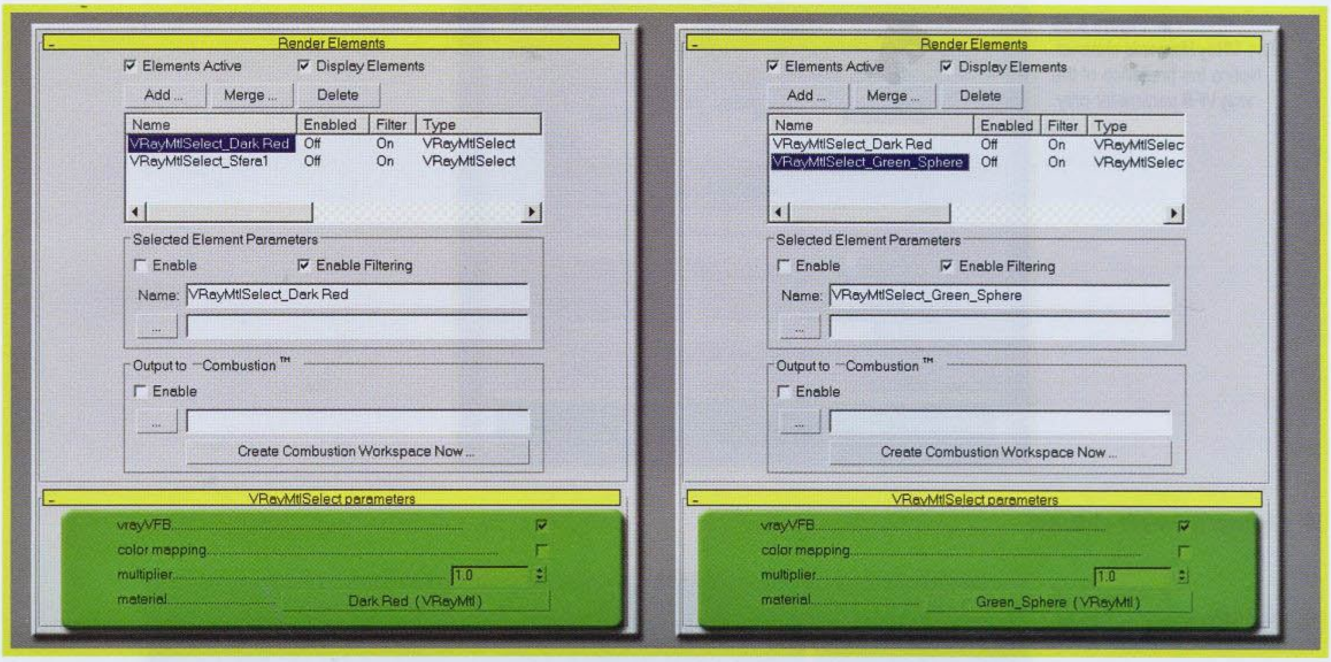
If one analyzes the generated image through the ***Pixel information*** tool, it discovers that color is not in ***RGB*** format, but is an integer. The number reflects the ***Material ID Channel***. Furthermore, no ***antialiasing*** is applied, despite it being active. This happens because the ID of a material, an integer, cannot be other than 3 or 4: fractional numbers cannot be present, therefore ***antialiasing*** cannot be computed. Even enabling the ***Enable Filtering*** option, no kind of ***antialiasing*** can be applied.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	no

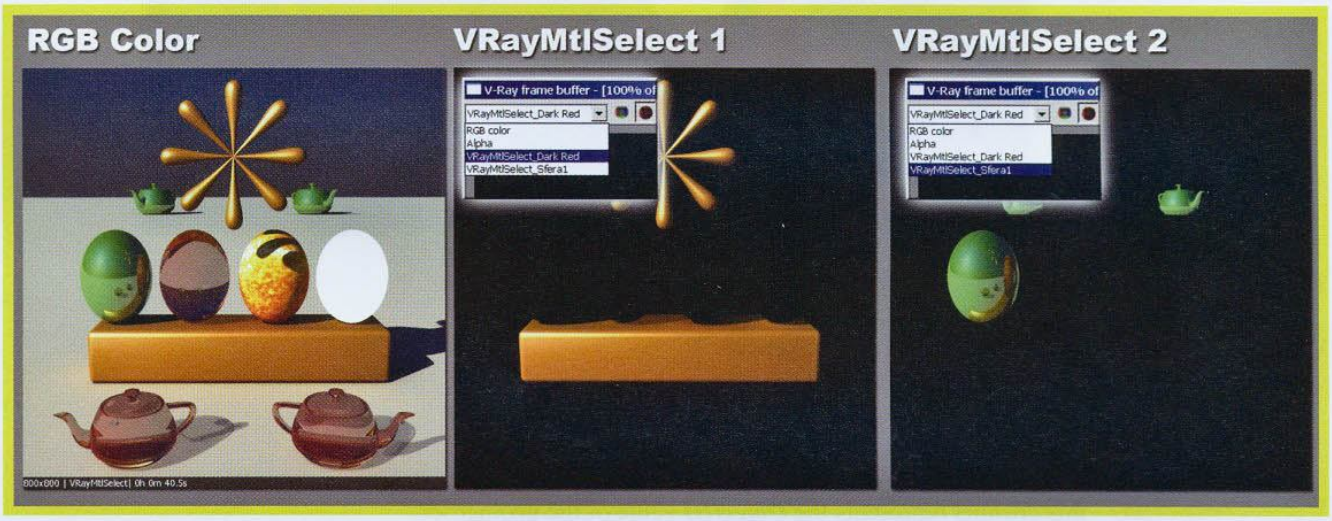
VRayMtlSelect - this channel allows one to isolate the objects according to the applied material. The result is an ***RGB Color*** channel, the complete render, for the objects with the selected material. The remaining areas will be completely black. It is possible to use more than one channel, one for each material.

In the next test two ***VRayMtlSelects*** are used, one for the orange-colored objects and the other for the green-colored objects.



■ **Figure 8.63**
Use of two ***VRayMtlSelect*** layers. Notice the introduction of a new parameter, with which it is possible to select the shader, and therefore the objects to be isolated.

The result of this channel:



■ **Figure 8.64**
In the pull-down menu there are as many layers as there are used ***VRayMtlSelect*** channels.

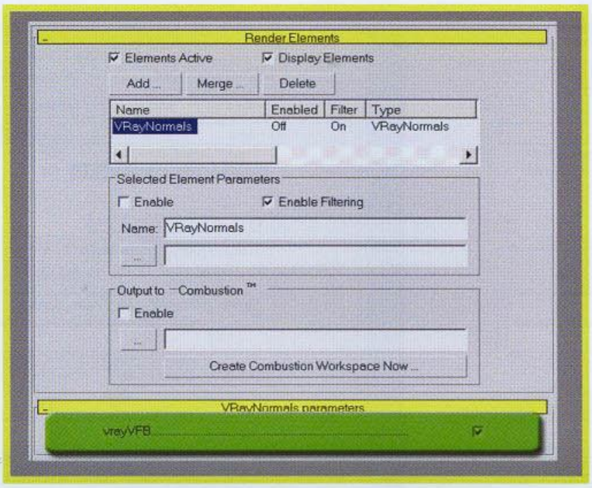
First of all one can see how two layers appear in the ***VRay Frame buffer***, one for each ***Render Element***. Furthermore, the isolation of the material is complete, comprehensive of all the elements: color, reflections, refractions, shadows, etc…

This channel supports:

VRayMtl	**3ds Max Mtl**	**Transparency**
yes	no	yes

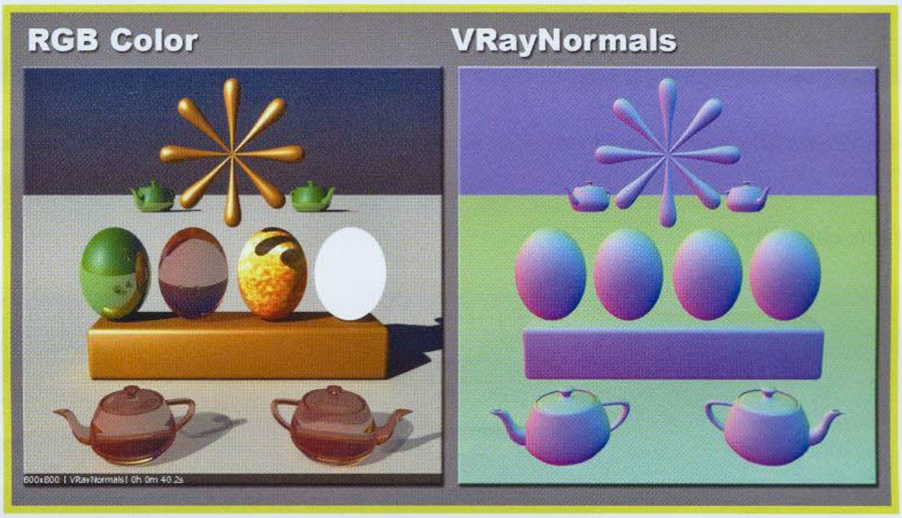
VRayNormals - this channel allows one to obtain an image representing the normals of the rendered objects.

■ **Figure 8.65**
The ***VRayNormals*** channel. Notice the presence of the ***vrayVFB*** parameter only.



The ***VRayNormals*** channel.

■ **Figure 8.66**
The ***VRayNormals*** channel.

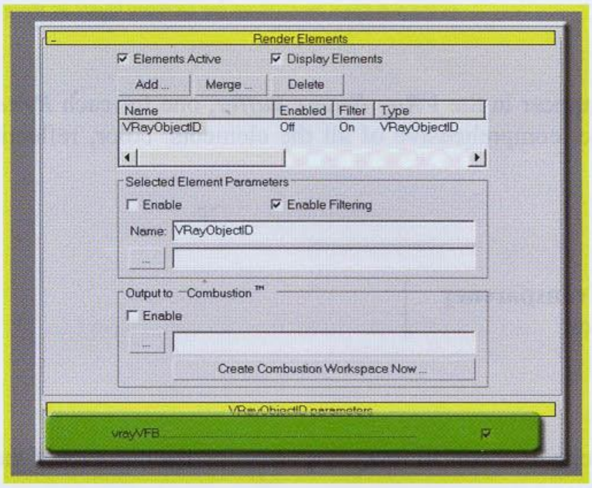


This channel supports:

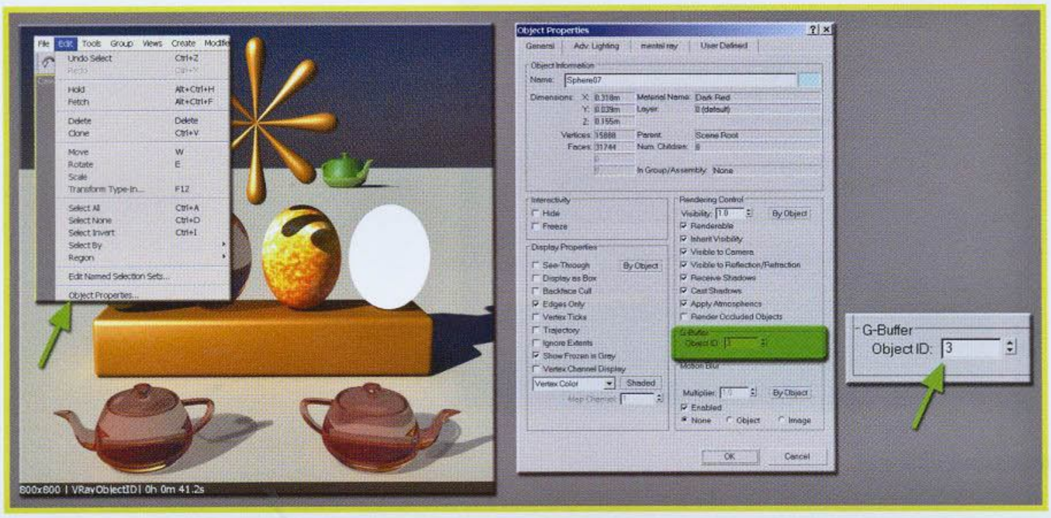
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayObject ID - this channel represents the ***ObjectID*** value of the geometries.

■ **Figure 8.67**
The ***VRayObjectID*** channel.

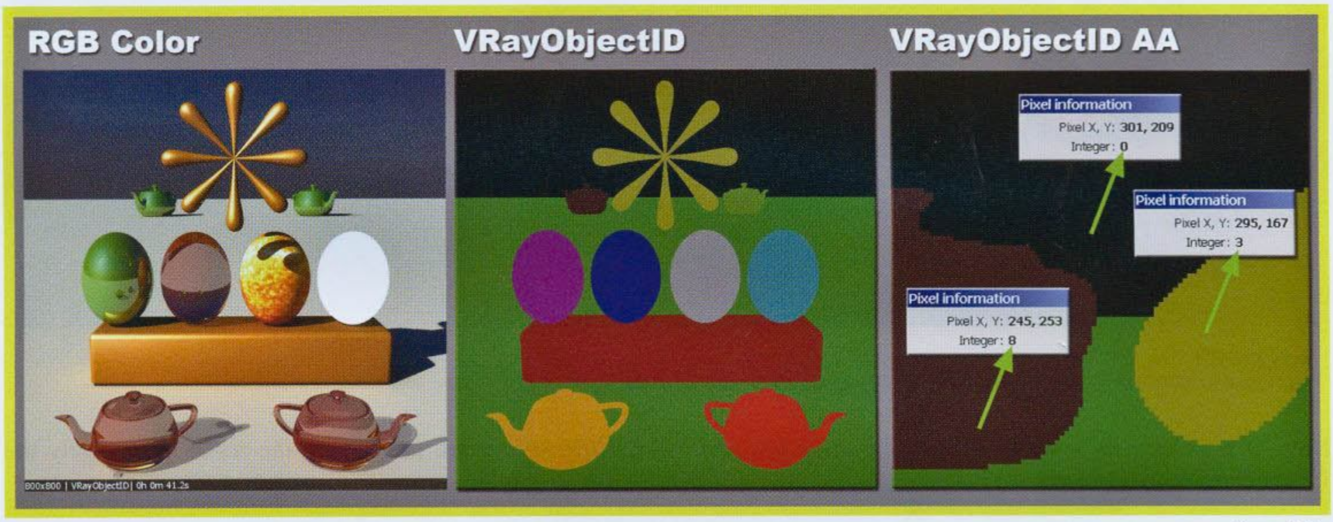


The *ObjectID* parameter, also called *G-Buffer*, is available by using the option Edit -> Object properties. The table has a section called *G-Buffer*. The integer assigned to the object is represented as color in the *VRayObjectID* channel.



■ **Figure 8.68**
A value of 3 for ***ObjectID*** has been assigned to the star-shaped object.

The result is a render colored by flat hues and integer numbers. This means that it is not possible to have fractional numbers, like one does with the *VRayMtlID*. The pixels represent 0 or 3. It is impossible to obtain an image with ***antialiasing***.



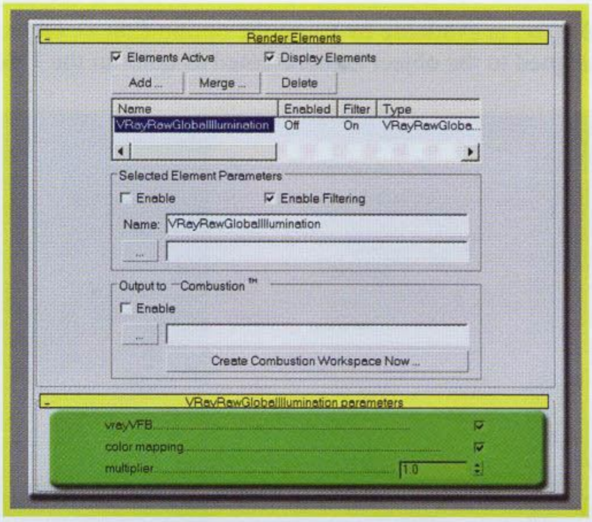
■ **Figure 8.69**
Employment of the parameter ***ObjectID*** for the generation of the ***VRayObjectID*** channel. The values are integers.

This channel supports:

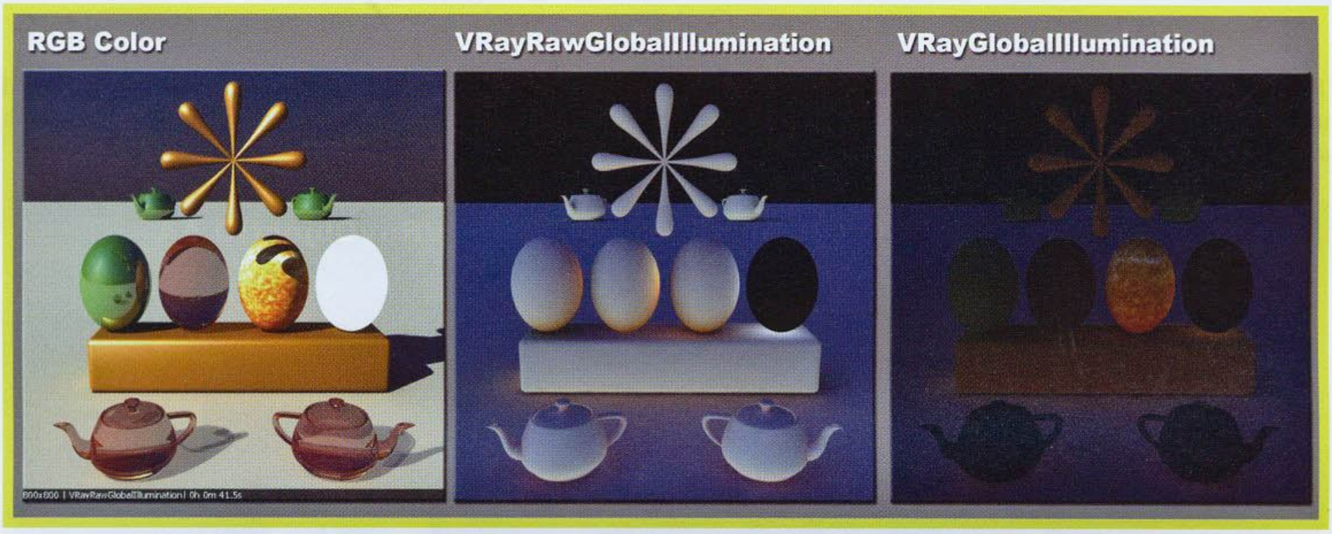
VRayMtl	3ds Max Mtl	Transparency
yes	yes	no

VRayRawGlobalIllumination - this channel represents the pure Global Illumination, without the multiplication for the *VRayDiffuseFilter* channel. This operation, in fact, generates the *VRayGlobalIllumination*. The parameters of the channel are the classic ***vrayVFB***, ***color mapping*** and ***multiplier*** in this case too.

■ Figure 8.70
The ***VRayRawGlobalIllumination*** channel.

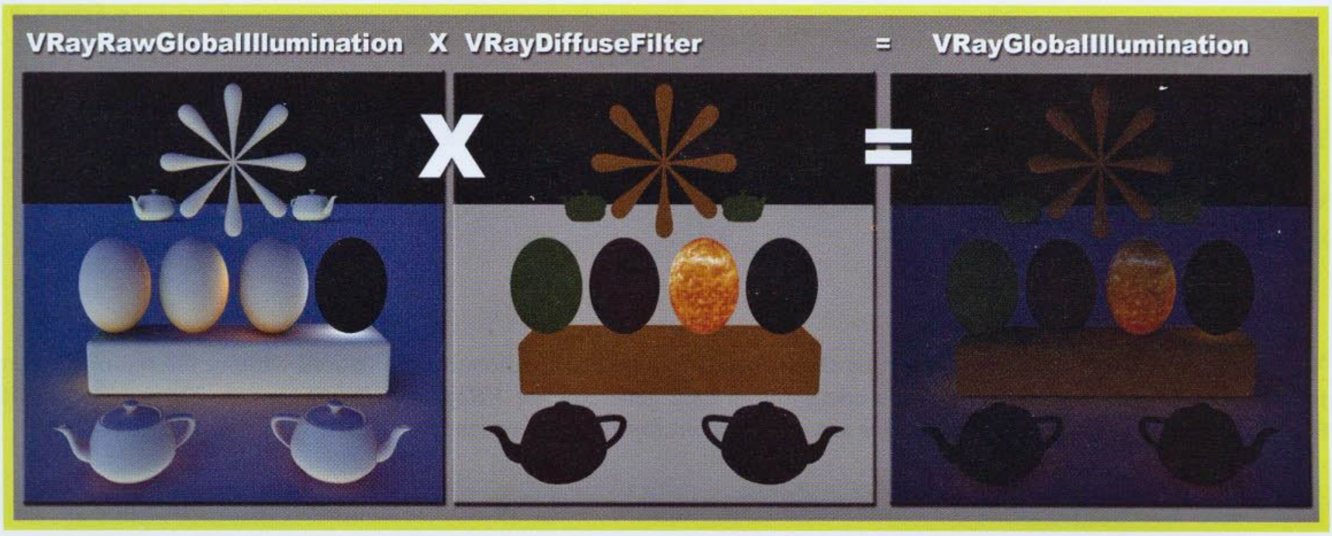


The ***VRayRawGlobalIllumination*** channel.



■ Figure 8.71
The ***VRayRawGlobalIllumination*** compared to the ***VRayGlobalIllumination*** channel.

Now, if the channels ***VRayRawGlobalIllumination*** and ***VRayDiffuseFilter*** are multiplied, the result is precisely the ***VRayGlobalIllumination***.

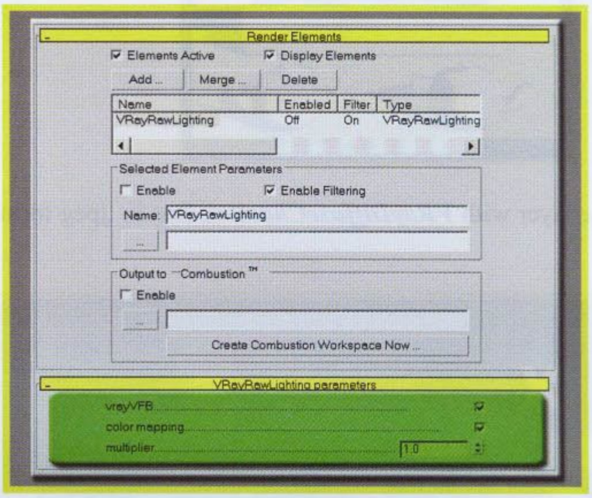


■ Figure 8.72
The multiplication of the channels ***VRayRawGlobalIllumination*** and ***VRayDiffuseFilter*** allows one to obtain the ***VRayGlobalIllumination*** channel.

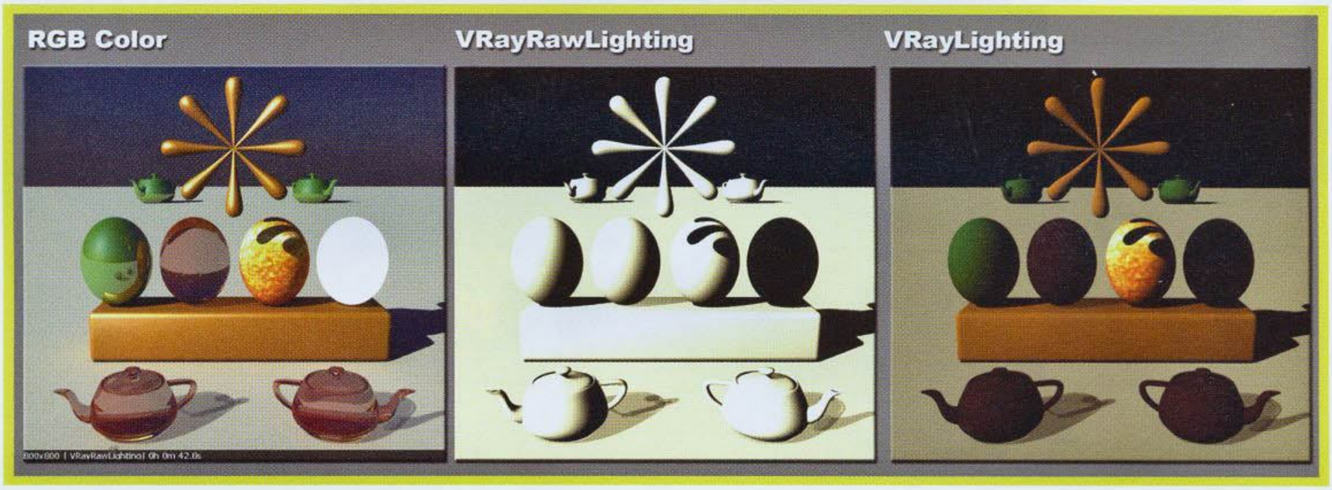
This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	no

VRayRawLighting - this channel represents only the direct illumination before the multiplication with the ***VRayDiffuseFilter*** channel. The multiplication of the latter to the ***VRayRawLighting*** gives birth to the ***VRayLighting***. In this case the parameters of the channel are the classic ***vrayVFB***, ***color mapping*** and ***multiplier***.

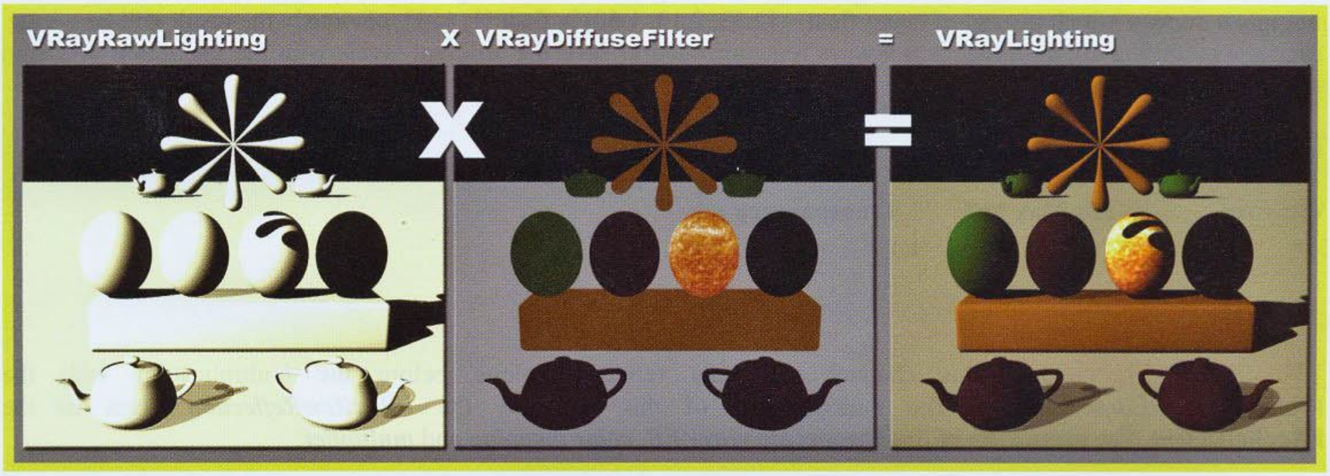


■ **Figure 8.73**
The ***VRayRawLighting***.



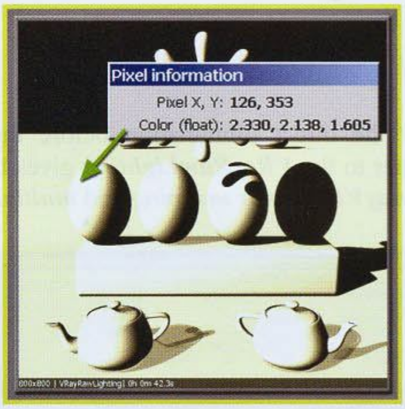
■ **Figure 8.74**
The ***VRayRawFilter*** channel. Notice the burning, due to values greater than RGB = 255.

The operation of multiplication between the channels ***VRayRawLighting*** and ***VRayDiffuseFilter***.



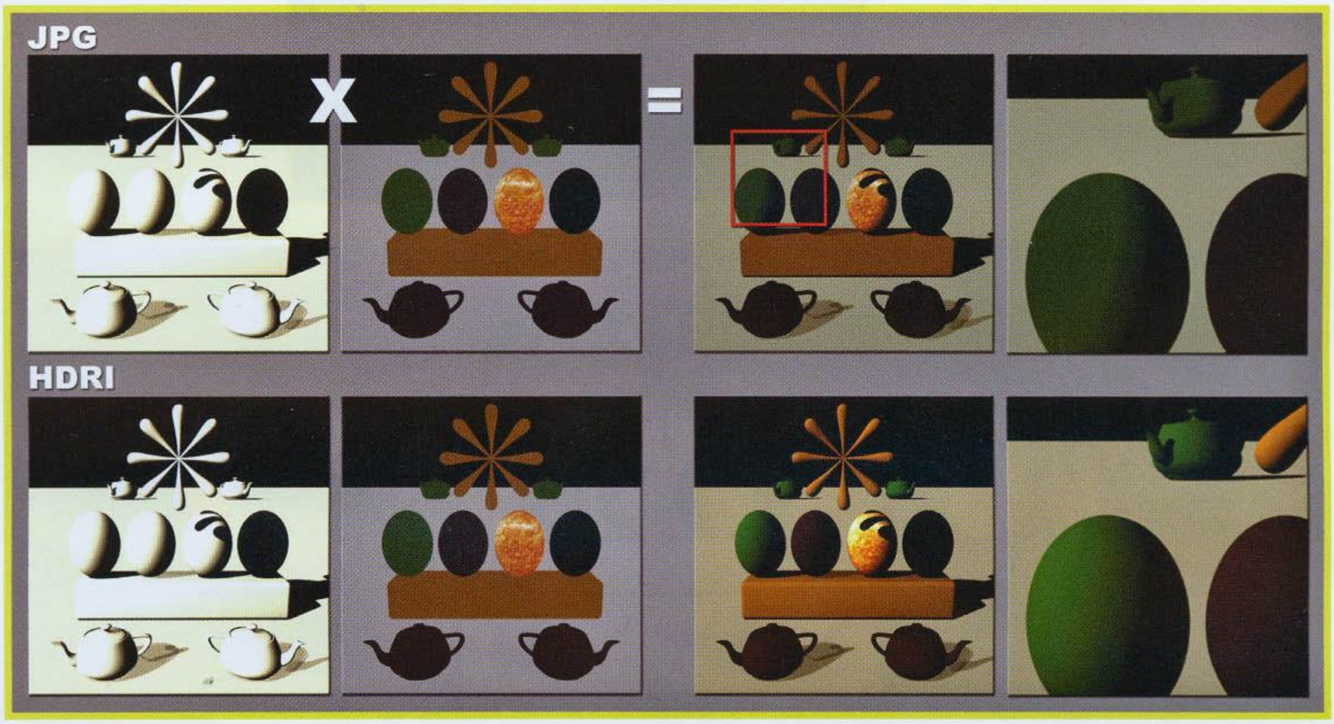
■ **Figure 8.75**
Multiplication between the two channels.

The file *VRayRawLighting* has values much higher than the float number = [1,1,1].



■ **Figure 8.76**
If the value is too high it makes the image appear "burned".

If one tries to multiply the layer with *VRayDiffuseChannel*, first in .jpeg format and than in .hdr, it is possible to obtain the following results:



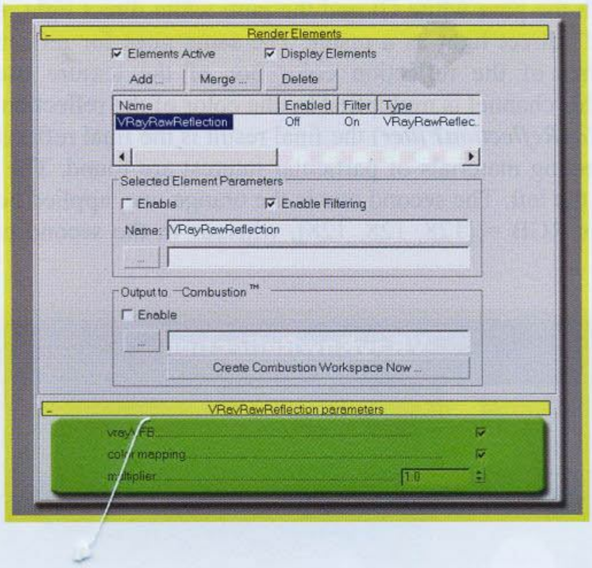
■ **Figure 8.77**
Multiplication between an 8 bit image and a floating point. Notice the green area of the LDRI image which is completely flat.

As explained in the preceding chapters, by using 8 bit images, the values higher than 1.0 are clipped. Having then to multiply a number which instead of being float = [2.33, 2.13, 1.60] is float [1, 1, 1], obviously the result will be wrong. That is why during compositing, it is always recommended to save the different levels with the maximum precision available, so in .exr or .hdr.

This channel supports:

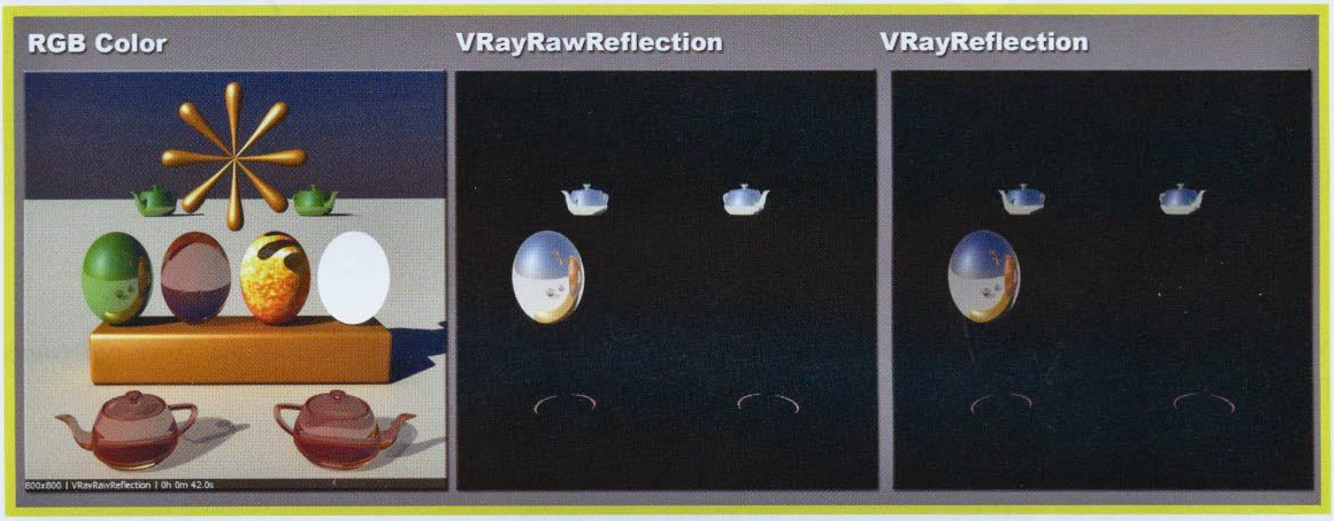
VRayMtl	**3ds Max Mtl**	**Transparency**
yes	no	yes

VRayRawReflection - this channel represents reflection alone before the multiplication with the *VRayReflectionFilter* channel. The multiplication of this layer by the *VRayRawReflection* gives us the *VRayReflection*. The parameters of this channel are *vrayVFB*, *color mapping* and *multiplier*.



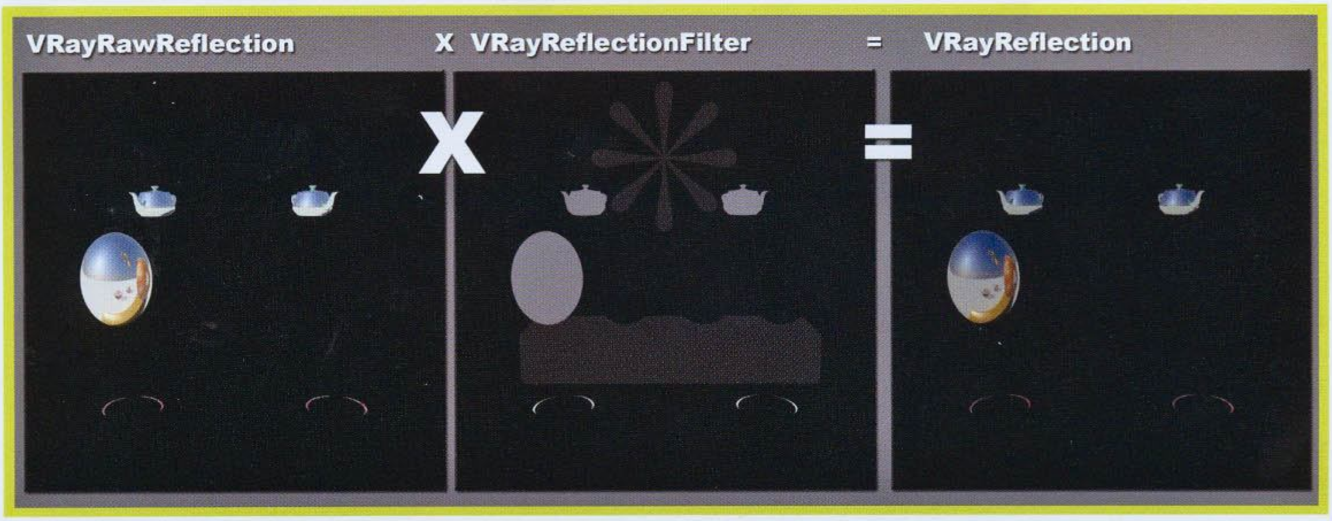
■ **Figure 8.78**
VRayRawReflection channel.

The channel ***VRayRawReflection***.



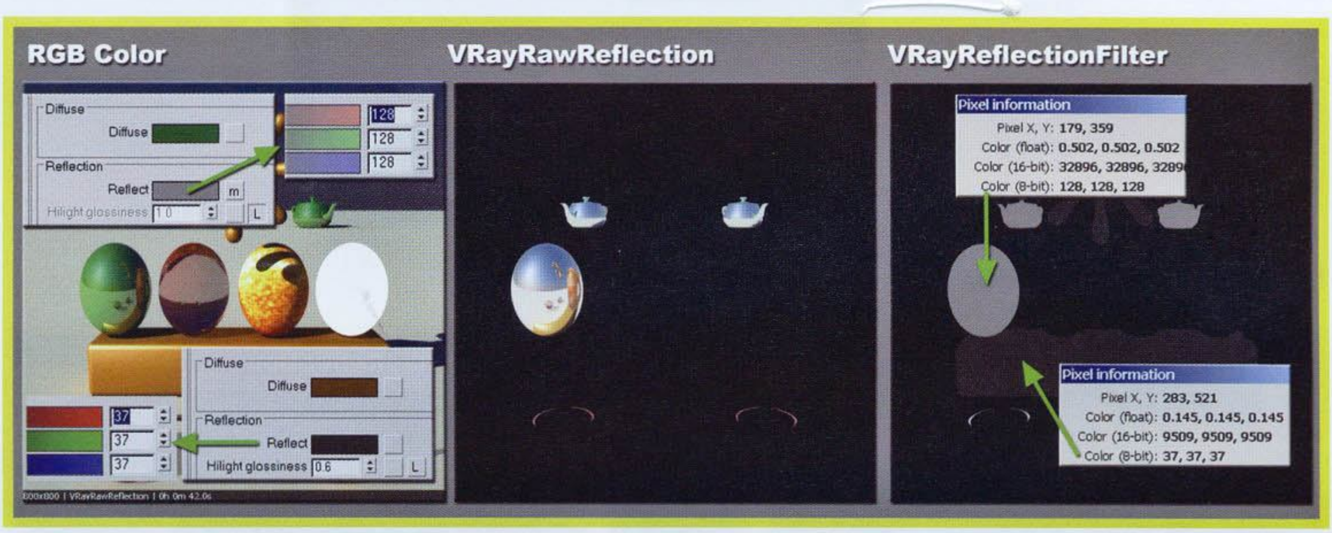
■ **Figure 8.79**
The two channels ***VRayRawReflection*** and ***VRayReflection***.

At first sight the two channels should seem identical, but if one pays attention, ***VRayReflection*** is slightly darker. The reason for this will become clearer as soon as the ***VRayReflectionFilter*** is analyzed.



■ **Figure 8.80**
The ***VRayReflectionFilter*** represents the color or the texture used in the reflection channel.

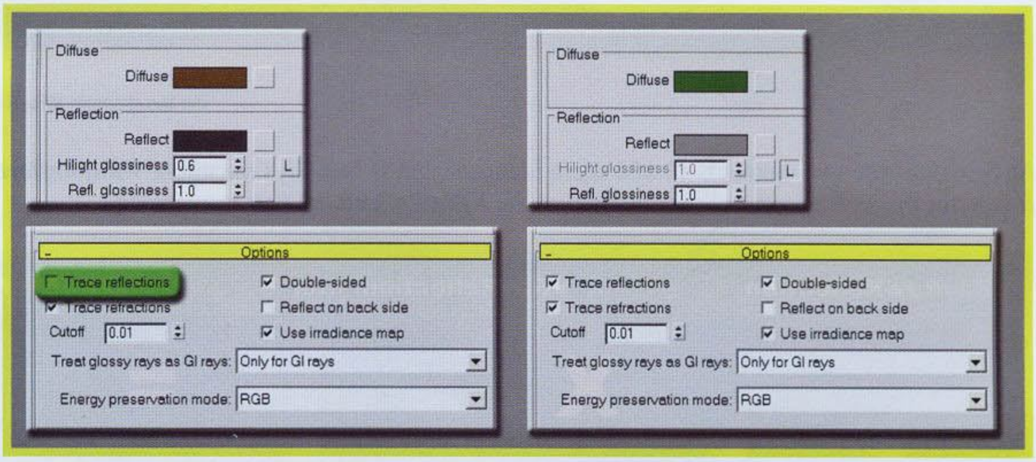
In the previous cases both the pure white GI and the pure white direct light, if multiplied by the surface color, give us "colored" GI or "colored" direct light as a result. The same goes for reflections. Pure reflections, meaning the ones without interference either of the reflection color nor of the texture used by the reflection, give birth to the ***VRayRawReflection***. If this channel is multiplied by the color of the reflection or by the texture present in the reflection (this channel is called ***VRayReflectionFilter***) the final result is the final reflection.
In this example two reflecting materials of particular interest are found. The first one is the green one, applied to the first sphere starting from the left. The second one is the orange one, applied both to the base and to the windmill. In the first case the reflection is RGB = [128, 128, 128], whereas in the second case it is RGB = [37, 37, 37], exactly as indicated in the VFB.



■ **Figure 8.81**
Notice the correspondence between the color of the reflection of the material and the ***VRayReflectionFilter***.

Why do neither the base or the windmill appear in the ***VRayRawReflection***? Simply because the ***Trace reflections*** parameter is disabled in ***VRayMtl*** options.

■ **Figure 8.82**
The deactivation of the parameter ***Trace reflections***, together with the use of the ***Highlight glossiness*** option, allows one to create high lights on the material, without generating reflections.

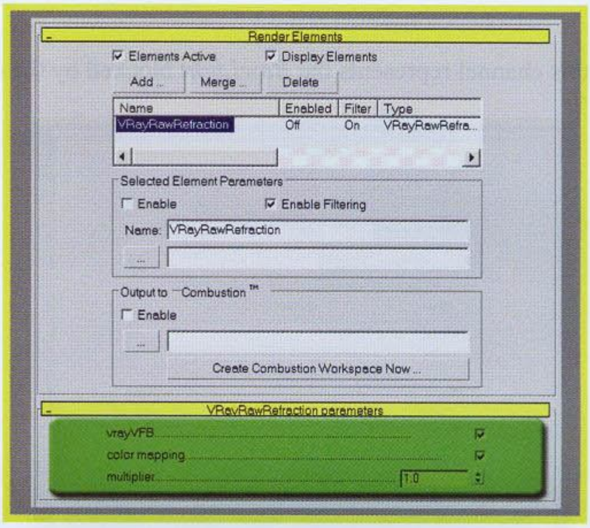


This way no reflection has been generated. The ***VRayRawReflection*** is, for those objects, completely black.

This channel supports:

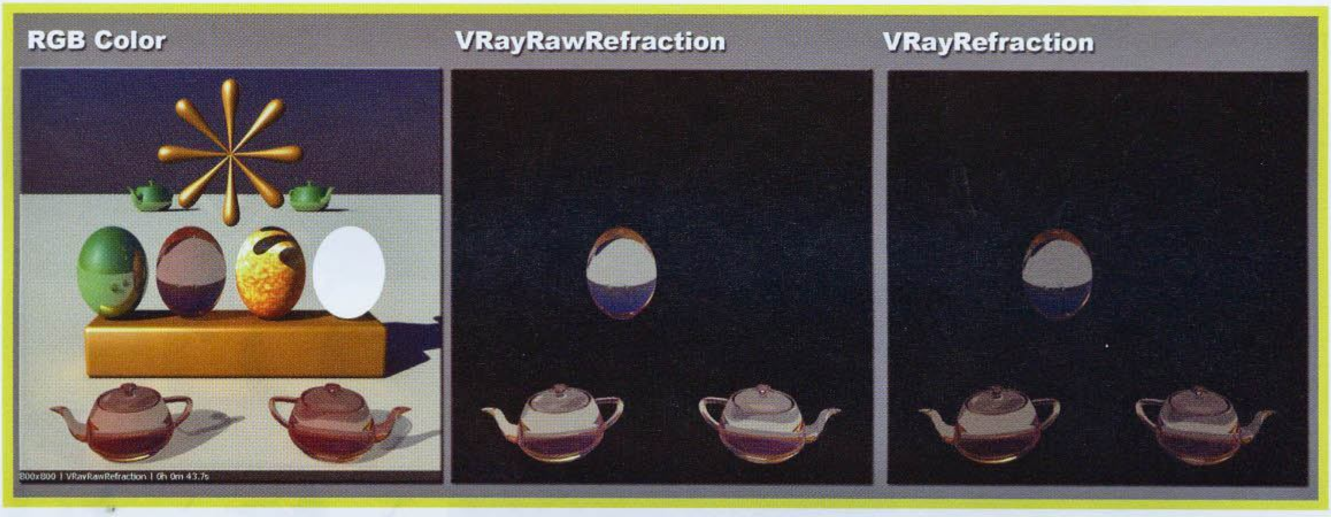
VRayMtl	**3ds Max Mtl**	**Transparency**
yes	no	yes

VRayRawRefraction - this channel represents the refraction only, before the multiplication by the ***VRayRefractionFilter*** channel. The multiplication of this layer by ***VRayRawRefraction*** generates ***VRayRefraction***, that is the final refractive surface. The available parameters of this channel are ***vrayVFB***, ***color mapping*** and ***multiplier***.



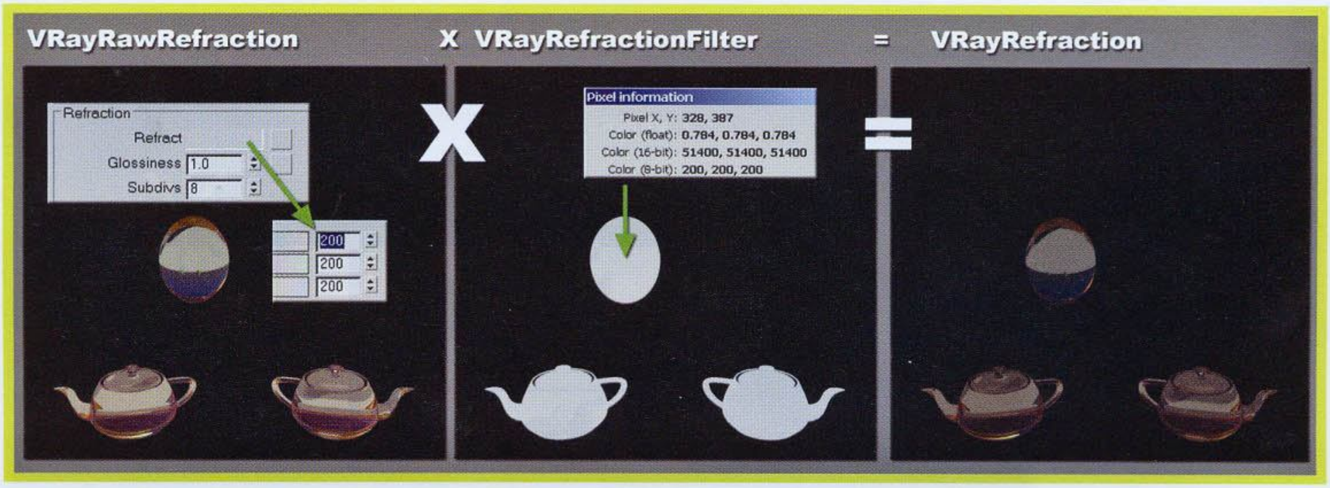
■ **Figure 8.83**
The ***VRayRawRefraction***.

The ***VRayRawRefraction*** channel.



■ **Figure 8.84**
Two of the three channels devoted to the refractions.

As for multiplication, review the observations remarked for the ***VRayRawReflection***. The same considerations are valid for ***VRayRawRefraction*** as well.

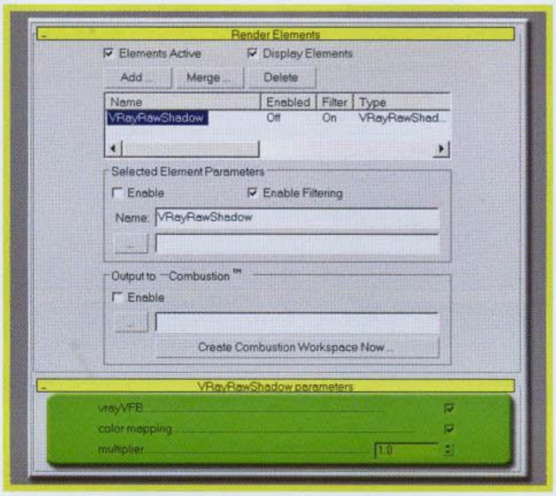


■ **Figure 8.85**
Both solid colors and textures can be put into the ***VRayRefractFilter***.

This channel supports:

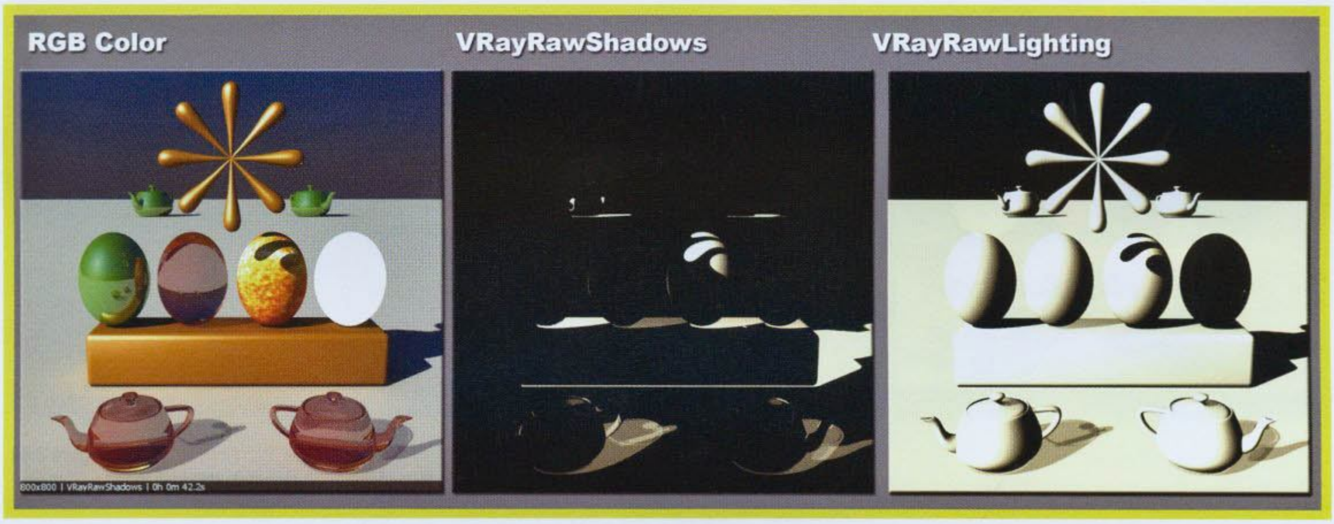
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayRawShadow - this channel represents the ***RawLight*** blocked by the objects.



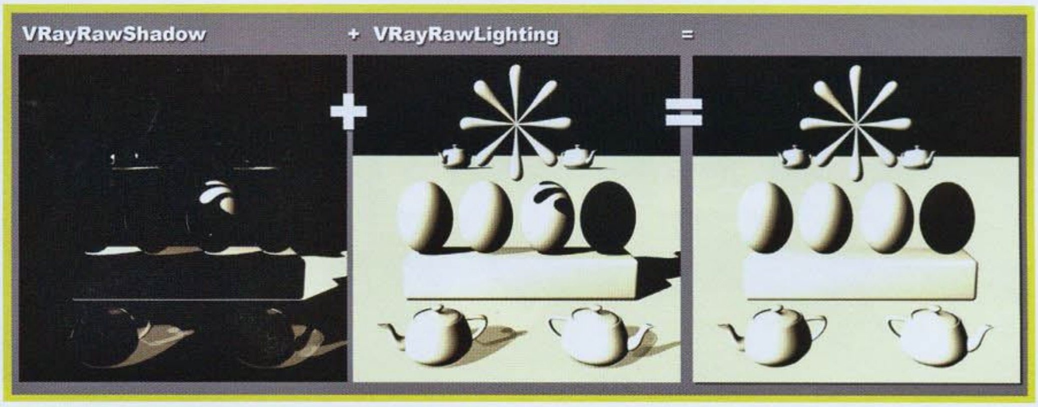
■ **Figure 8.86**
Like in the previous cases, also in this case there are three available parameters in the ***VRayRawShadow*** channel.

It is interesting to notice how the ***VRayRawShadow*** channel is exactly the opposite of the ***VRayRawLighting*** one.



■ **Figure 8.87**
The ***VRayRawShadow*** channel is exactly the opposite of the ***VRayLighting*** one.

In fact, by adding the ***VRayRawShadow*** to ***VRayRawLighting***, the result is the removal of all shadows.

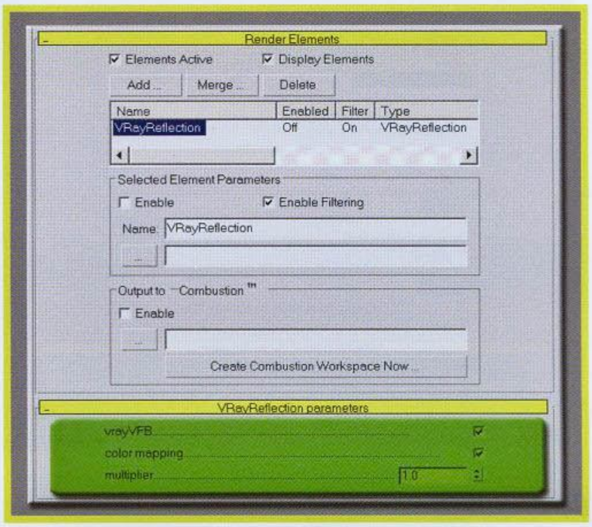


■ **Figure 8.88**
Algebraic sum of the two layers.

This channel supports:

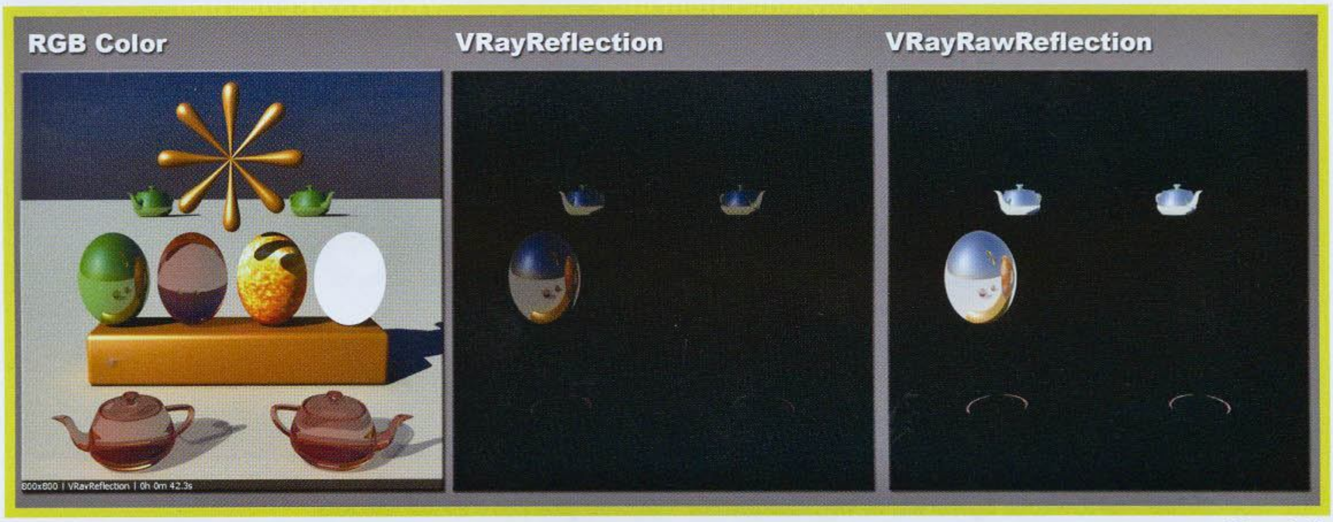
VRayMtl	3ds Max Mtl	Trasparenza
yes	no	yes

VRayReflection - we have already mentioned this channel during the explanation of the ***VRayRawReflection*** channel. The ***VRayReflection*** channel represents the complete surface reflection. It is possible to obtain this channel both by means of simple ***Render Elements*** or through the multiplication of the ***VRayRawReflection*** with the ***VRayReflectionFilter***.



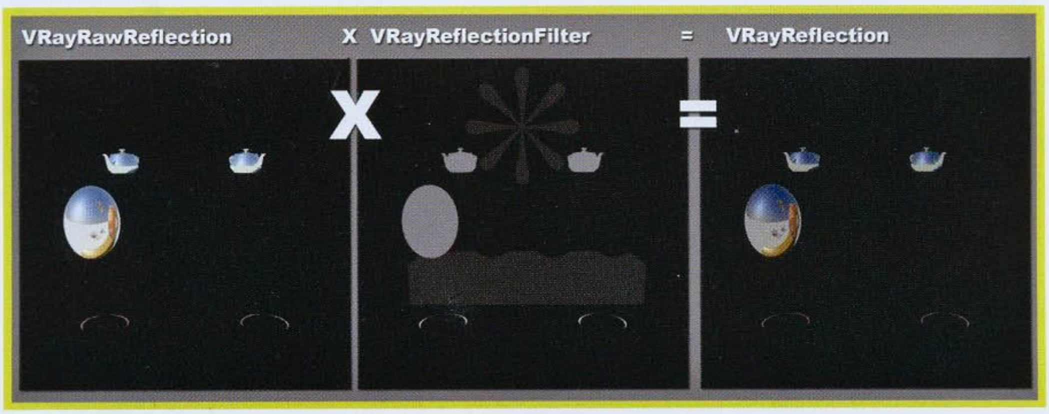
■ **Figure 8.89**
VRayReflection. There are three parameters in the channel.

The ***VRayReflection*** channel.



■ **Figure 8.90**
Comparison between *VRayReflection* and *VRayRawReflection*.

For a complete analysis of the channel go back to the ***VRayRawReflection*** layer.

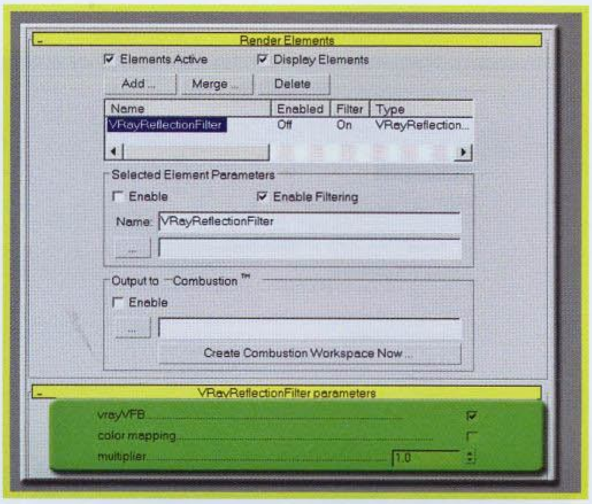


■ **Figure 8.91**
The *VRayReflection* channel.

This channel supports:

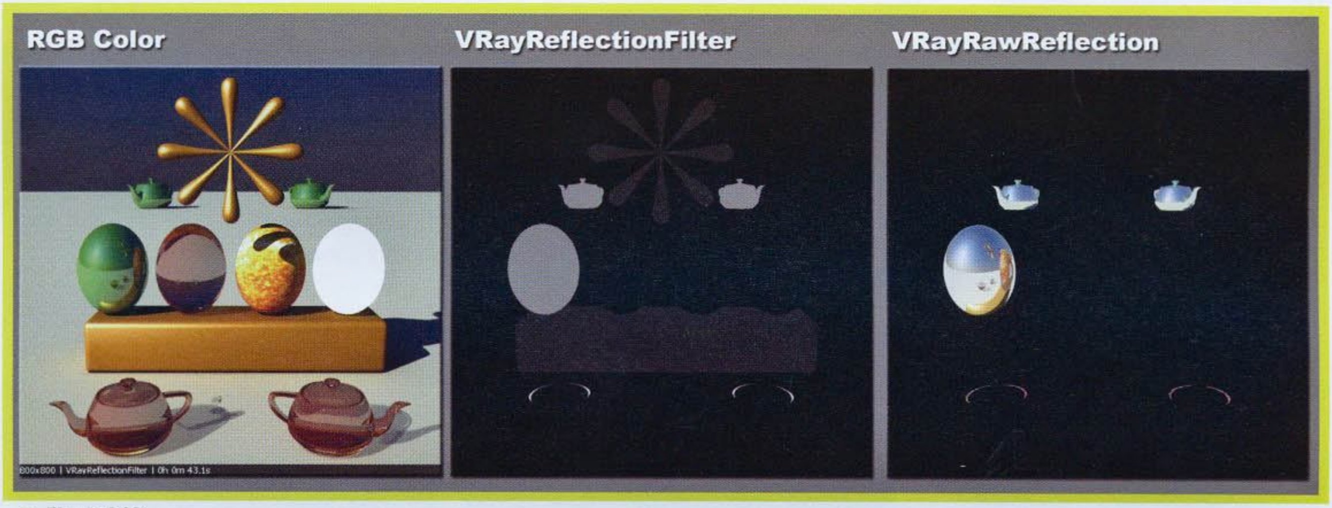
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayReflectionFilter - this layer represents the "color" of the reflection channel. By multiplying it by ***VRayRawReflection*** the result is the final reflection, that is, ***VRayReflection***. For more details see ***VRayRawReflection***.



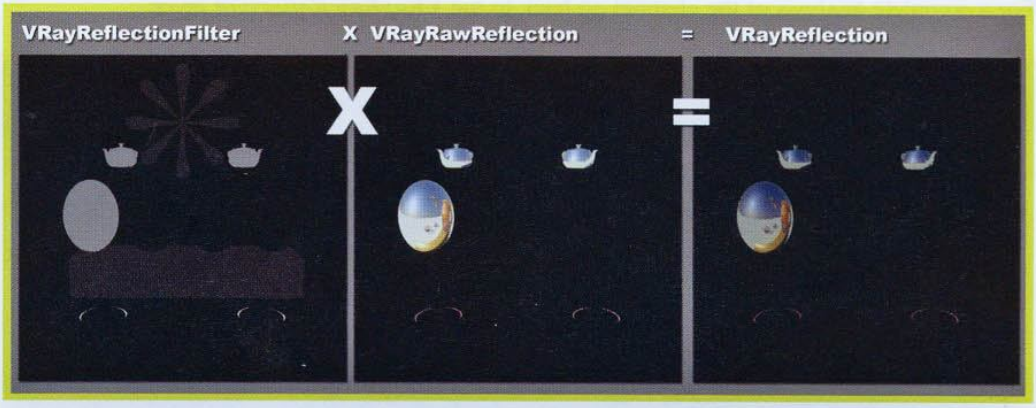
■ **Figure 8.92**
The three classic parameters for controlling the channel.

The well-known ***VRayReflectionFilter***.



■ **Figure 8.93**
The ***VRayReflectionFilter*** contributes, together with the ***VRayRawReflection***, towards the generation of the final reflections.

The multiplication with the ***VRayRawReflection***.

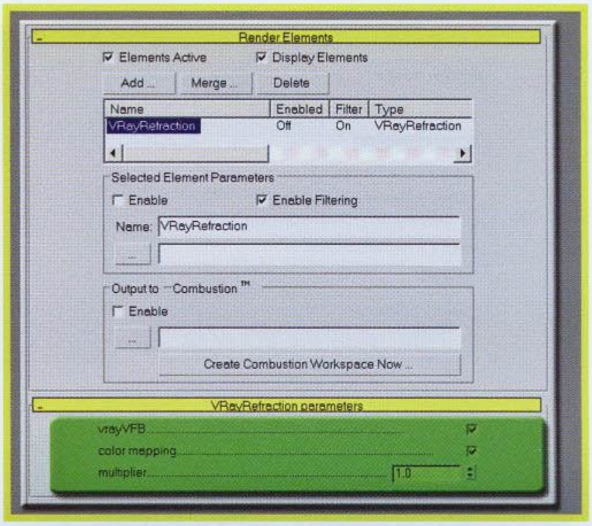


■ **Figure 8.94**
Algebraic multiplication of the two channels responsible for the final reflections.

This channel supports:

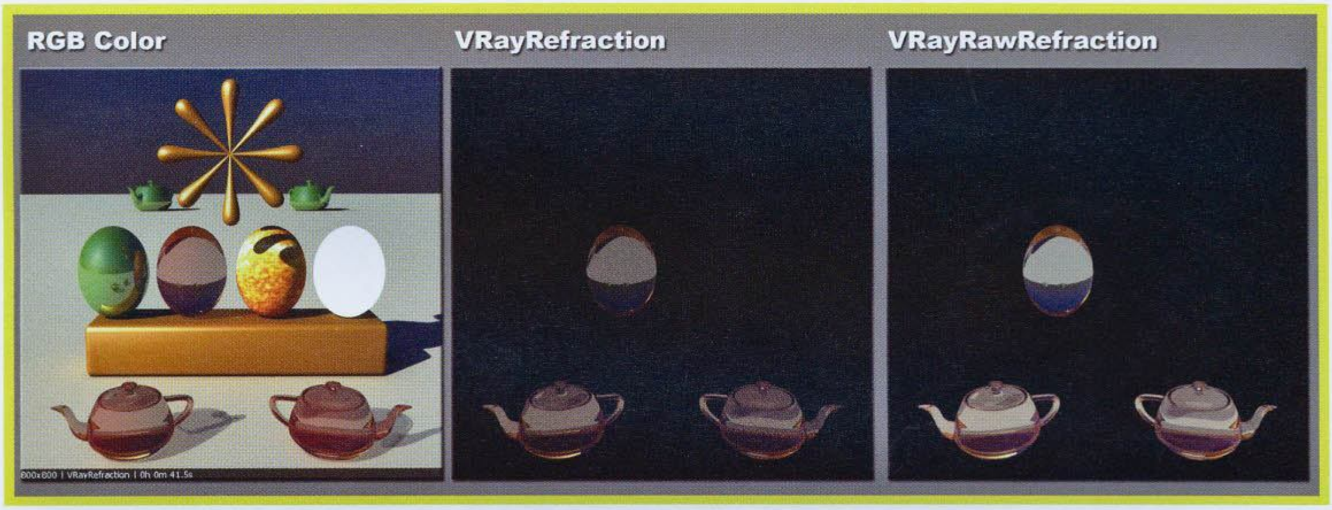
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayRefraction - this channel represents the complete refractions found in the render.



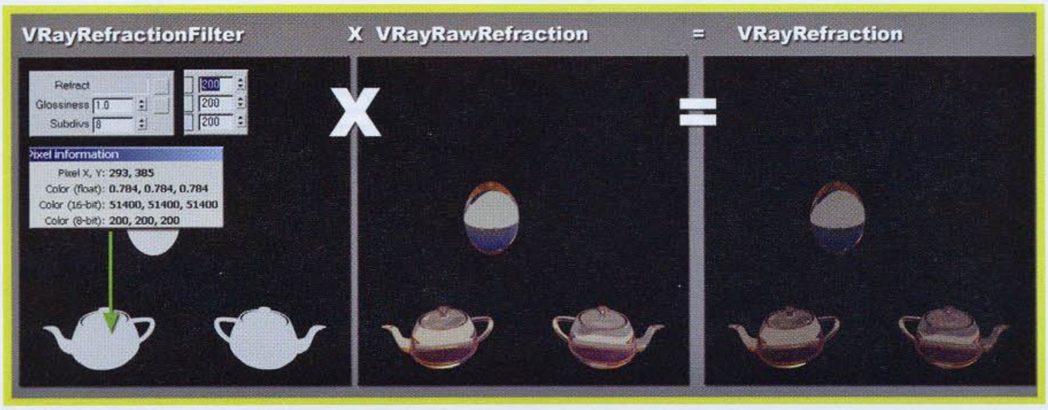
■ **Figure 8.95**
The ***VRayRefraction***. There are three parameters in the channel.

The final refractions present in the ***VRayRefraction*** channel.



■ **Figure 8.96**
Comparison between ***VRayRefraction*** and ***VRayRawRefraction***.

This channel can be obtained from the multiplication of ***VRayRawRefraction*** with ***VRayRefractionFilter*** as well, just like reflections.

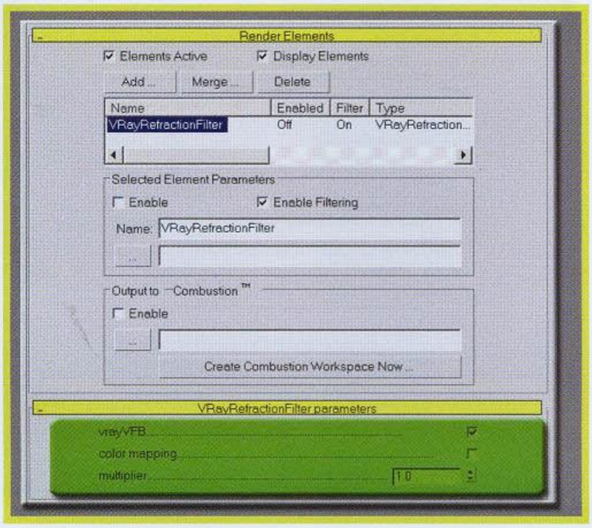


■ **Figure 8.97**
Algebraic multiplication of the two channels responsible for the final refractions.

This channel supports:

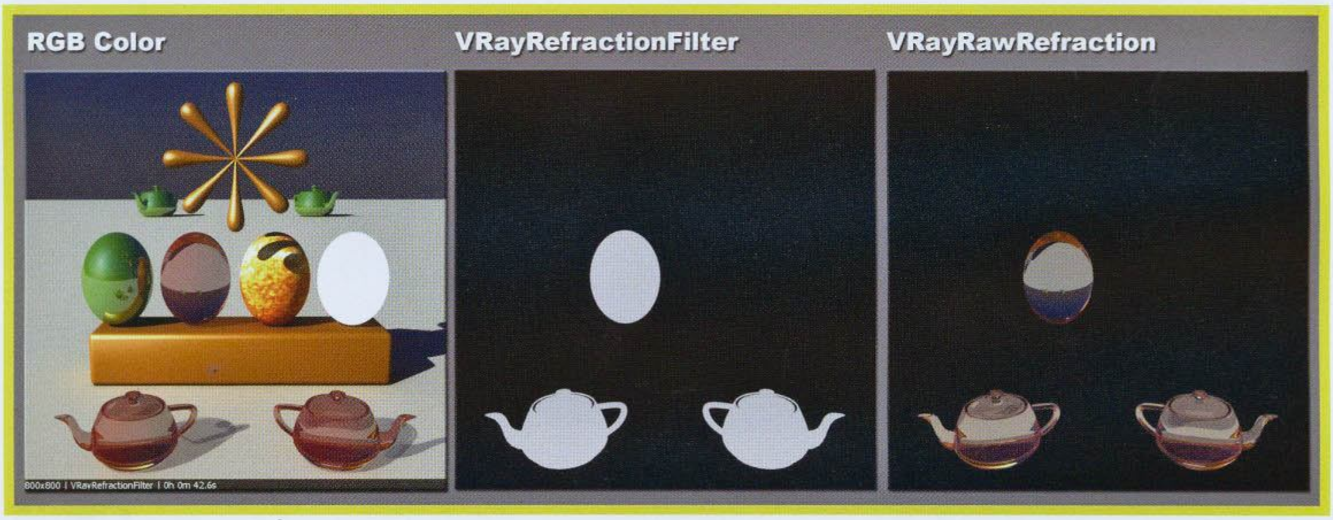
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayRefractionFilter - this layer represents the "color" of the refraction channel. By multiplying it with ***VRayRawRefraction*** the result is the total refraction, namely ***VRayRefraction***.



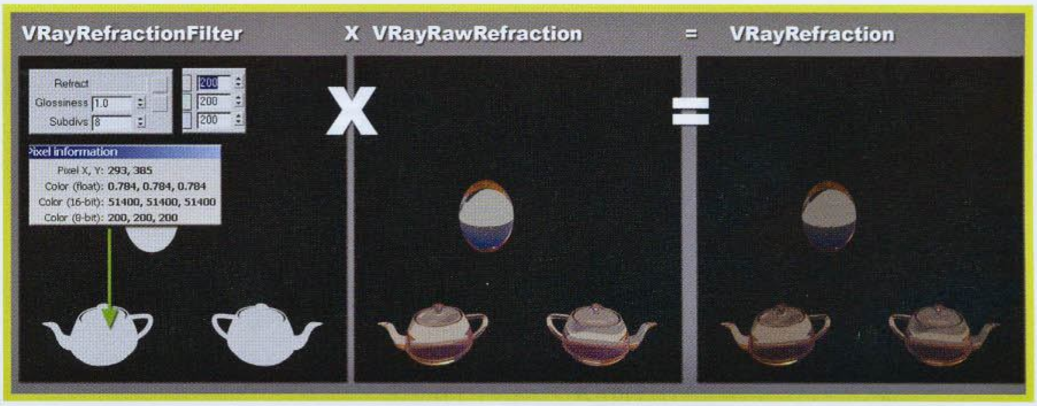
■ **Figure 8.98**
The three classic parameters for controlling the ***VRayRefractionFilter*** channel.

The ***VRayRefractionFilter***.



■ **Figure 8.99**
VRayRefractionFilter contributes, together with ***VRayRawRefraction***, to the generation of the final reflections.

The multiplication with the ***VRayRawRefraction*** and the***VRayRefractionFilter***.

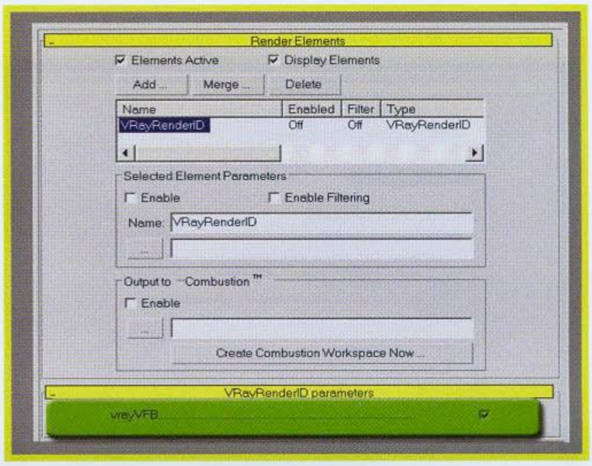


■ **Figure 8.100**
Algebraic multiplication of the two channels.

This channel supports:

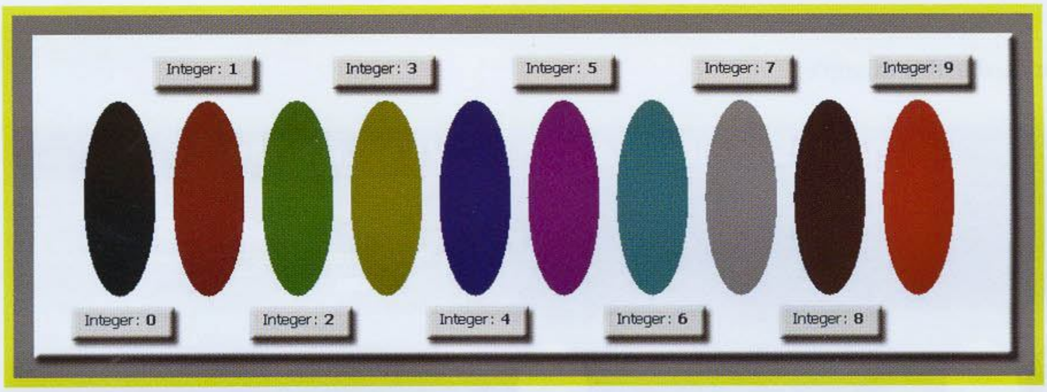
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayRenderID - each object has an enumeration identifying it compared to the other ones according to the order of creation. As a consequence, the first object which is created is number 1, the second number 2, etc... *VRay* , together with the ***VRayRenderID***, generates an image with integers which identifies this order.



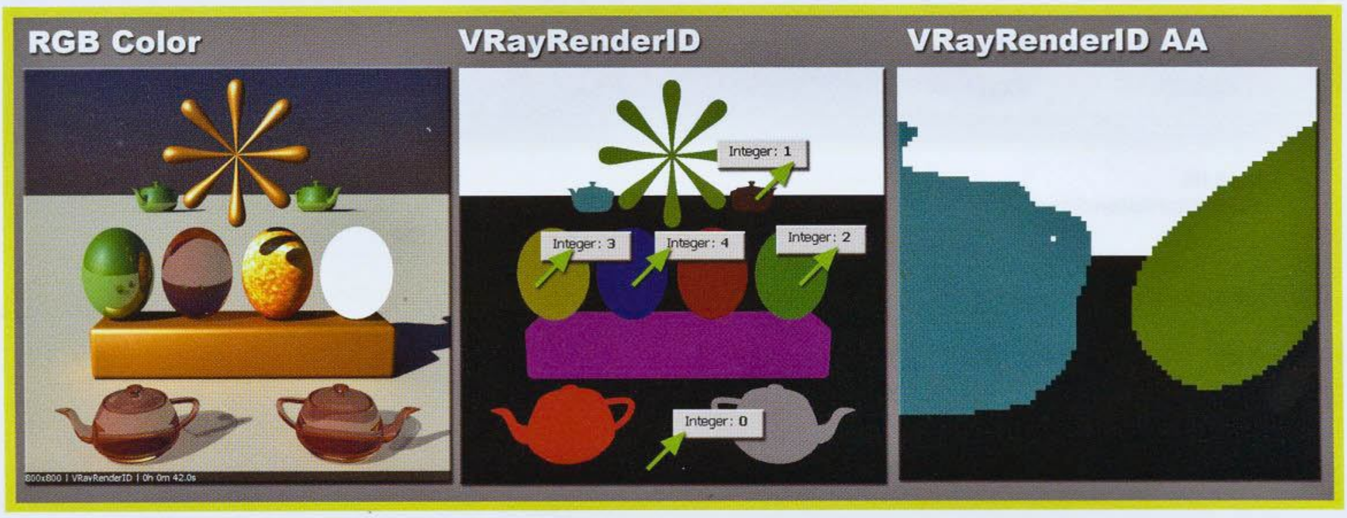
■ **Figure 8.101**
The ***VRayRenderID*** channel. In this case only one parameter exists. In fact, this layer can represent only integers and is not affected by lights, shadows nor by the GI, just like what happens for the ***VRayMtl*** and the ***VRayObjectID***.

In the following image some geometries, generated from left to right, are rendered so as to clarify how *VRay* uses colors. The first geometry is number 0 with a color which corresponds to RGB=[0, 0, 0]; the second one is number 2 and the color is therefore RGB = [128, 0, 0] ...



■ **Figure 8.102**
Some values generated by the ***VRayRenderID*** channel.

The ***VRayRenderID*** layer does not have ***antialiasing***, just like the ***VRayMtlID*** and the ***VRayObjectID*** channels.



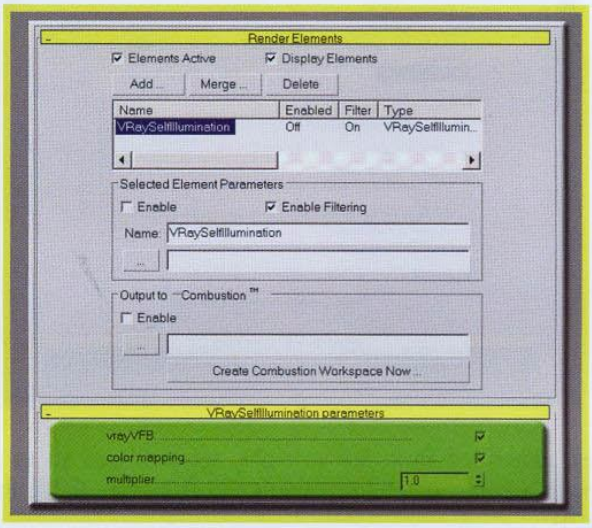
■ **Figure 8.103**
Integer colors represent the ***RenderID***. It is not possible to have ***antialiasing***.

This channel supports:

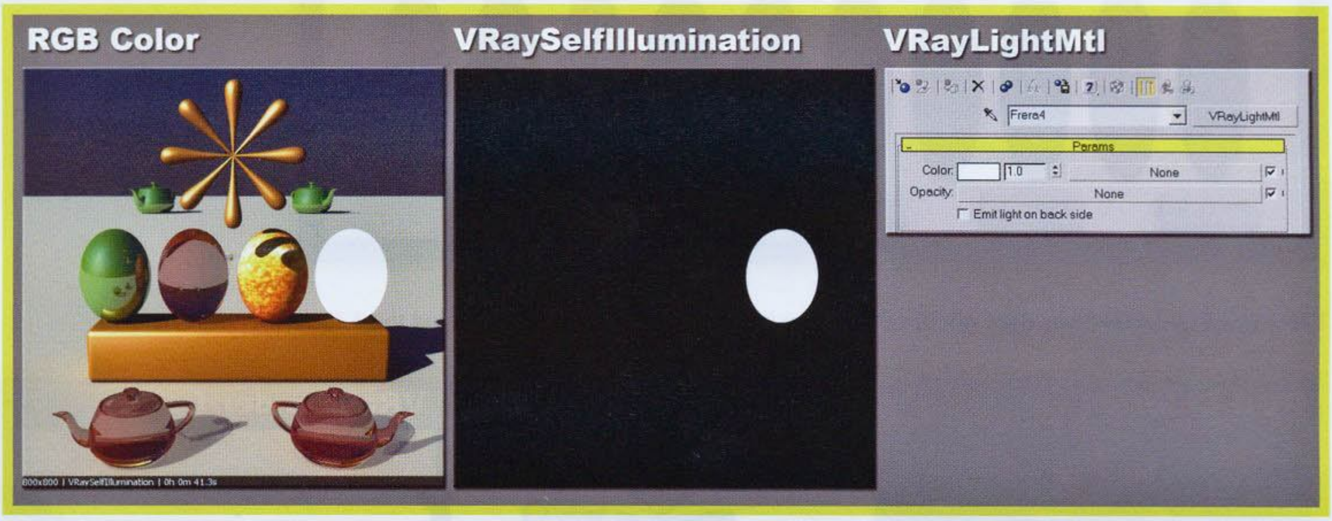
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRaySelfIllumination - this channel represents self-illuminating geometries, that is the meshes to which a ***VRayLightMtl*** has been assigned or where a Standard material with the ***self-illumination*** channel active is used.

■ Figure 8.104
The three parameters for controlling the *VRaySelfIllumination*.



A ***VRayLightMtl*** has been used in this example.

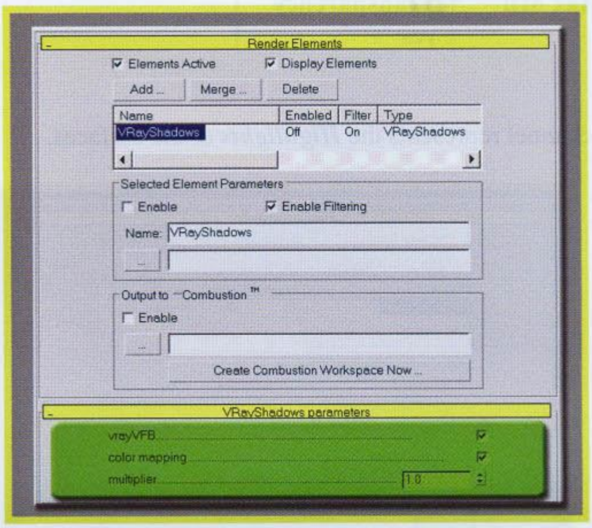


■ Figure 8.105
The *VRaySelfIllumination* channel.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayShadows - this channel represents the diffuse light blocked by the objects in the scene.

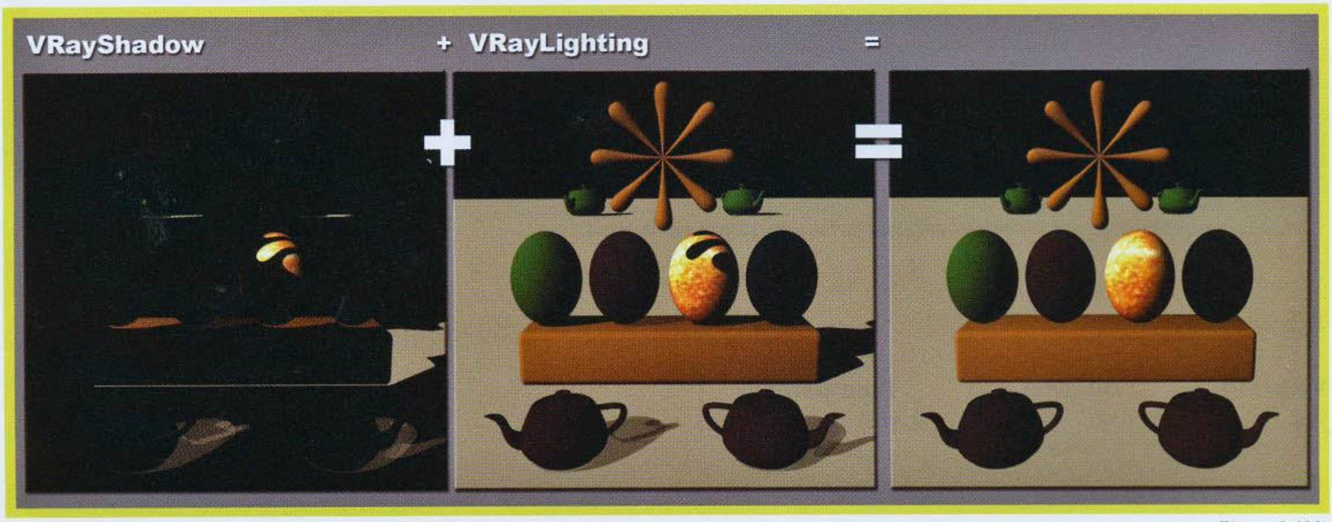


■ **Figure 8.106**
The ***VRayShadows*** channel and its parameters.

Comparison between the ***VRayShadows*** and the ***VRayLighting***.

■ **Figure 8.107**
Notice the perfect inverse correspondence between the two channels.

By summing the ***VRayShadow*** with the ***VRayLighting***, the result is the nulling of any direct shadow.

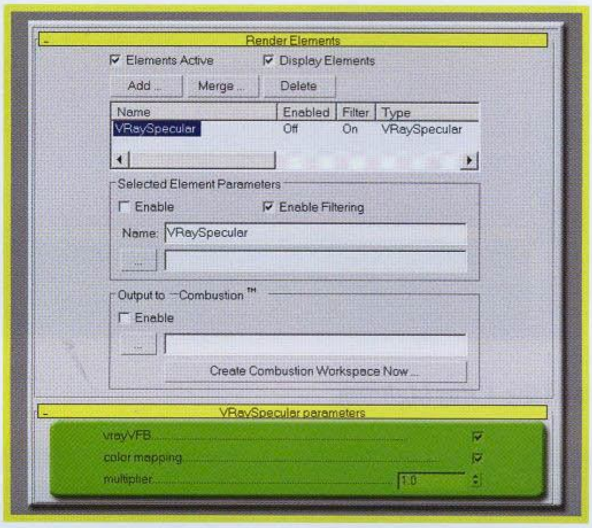


■ **Figure 8.108**
Deleting the shadows by using mathematical sums of the two channels.

This channel supports:

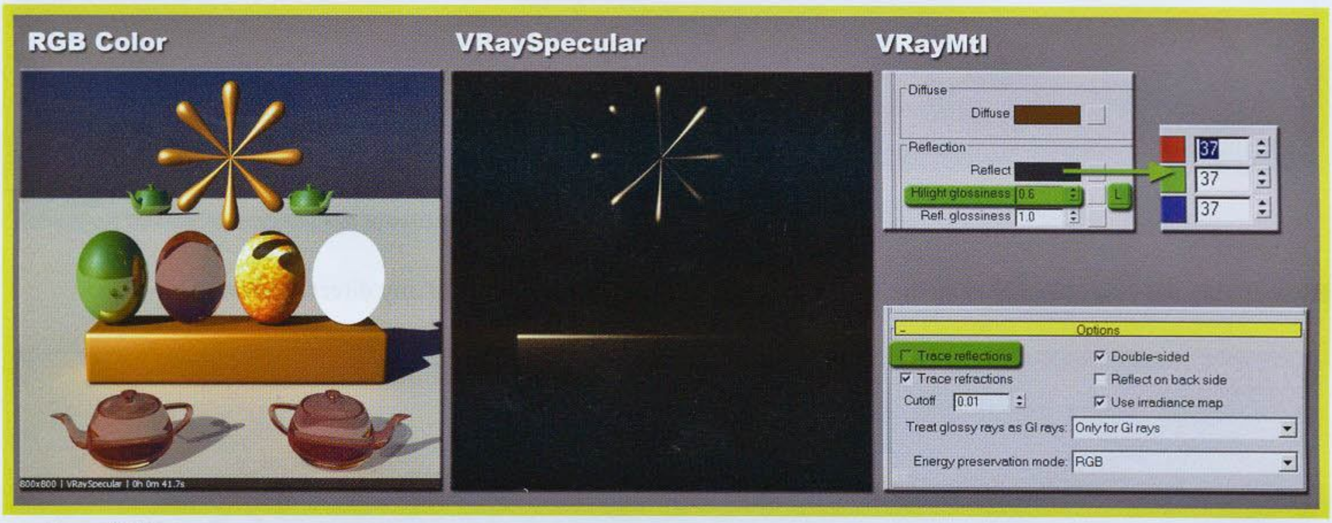
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRaySpecular - this channel represents the ***Highlights*** of the surfaces.



■ **Figure 8.109**
The three parameters for controlling ***VRaySpecular***.

Highlights can be generated in two different ways: by using the Standard material of 3ds Max or by modifying the parameters controlling the appearance of highlights, or by using the ***VRayMtl***, by using reflections, but deactivating their tracing. For more details, it is suggested to read the chapter concerning the ***VRayMtl***.
In this specific case a ***VRayMtl*** material with the reflections activated but the tracing deactivated has been used for the base and the windmill.

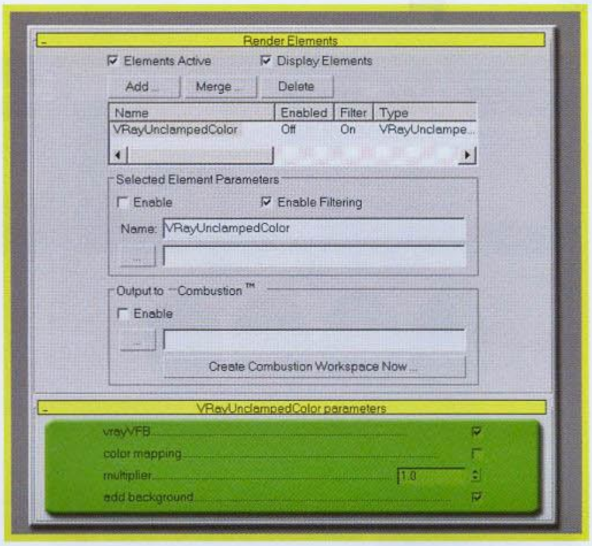


■ **Figure 8.110**
The ***VRaySpecular*** channel and the material employed for the generation of the ***Highlight***.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

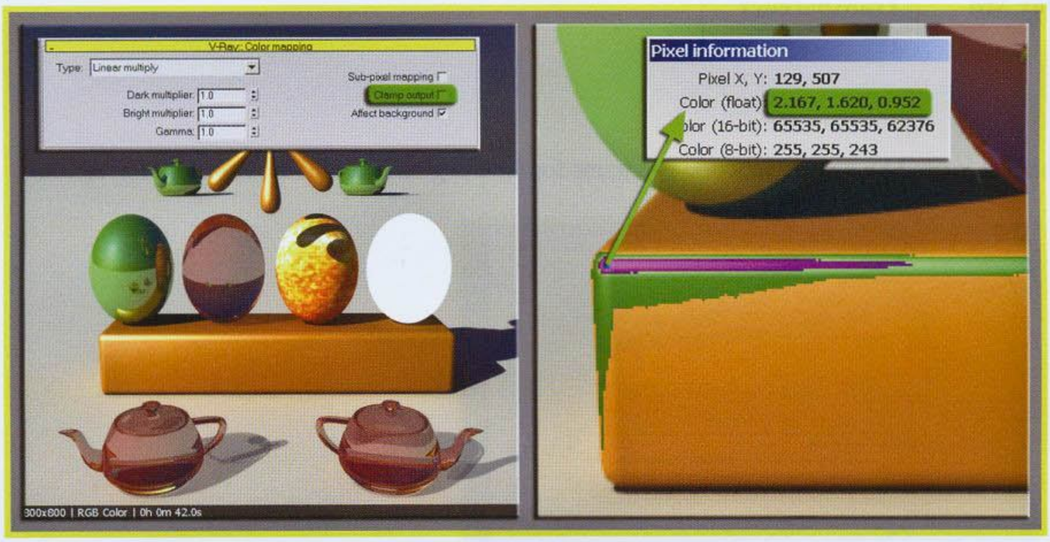
VrayUnclampedColor - this channel represents the "*unclamped*" colors of the render.



■ **Figure 8.111**
The ***VRayUnclampedColor*** and its parameters. In this case an additional option is present: the ***add background*** parameter. It allows one to establish whether the channel is going to render the background or not. If it is deactivated, the background is black.

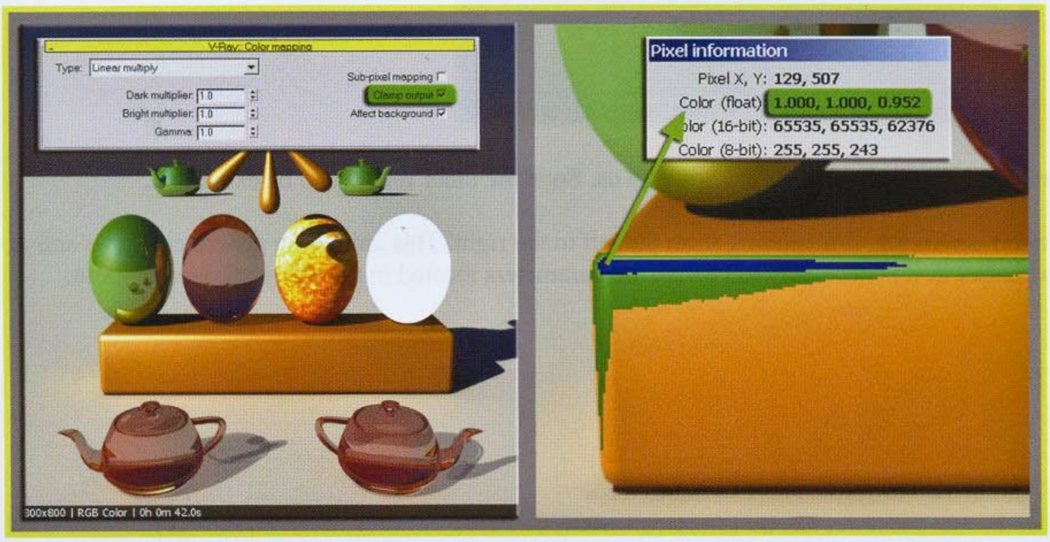
So far all the renders and the channels have been computed by using a linear-type ***Color mapping***, without changing the gamma or anything else. The ***Clamp output*** parameter has always been deactivated as well. This way one is able to save the generated images in float numbers without any kind of "clamp" over the value 1.0. By using the ***Pixel Information*** tool it is possible to know the exact value of a pixel.

.



■ **Figure 8.112**
By deactivating the ***Clamp output*** option it is possible to read the real value of the pixel in float numbers.

Instead, by activating the ***Clamp output*** option, *VRay* "clips" all the values exceeding 1.0. This way one obtains a normal 8-bit image.



■ **Figure 8.113**
Activation of the ***Clamp output*** option. All the values of the pixels exceeding 1.0 are "clamped".

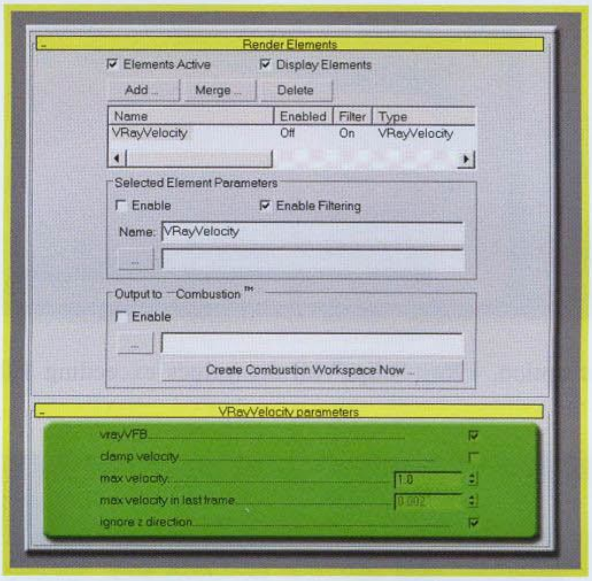
By using the ***VRayUnclampedColor*** channel, with the activation of the ***Clamp output*** option as well, one makes sure that the channel is created in float numbers without any kind of "clamping" of pixels, meaning without the influence of the ***Clamp output*** parameter.

■ **Figure 8.114**
The ***VRayUnclampedColor*** allows one to generate a render in float numbers, with the ***Clamp output*** parameter being activated too.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayVelocity - this channel is used by post-production programs for the 2D generation of the ***Motion blur*** effect.



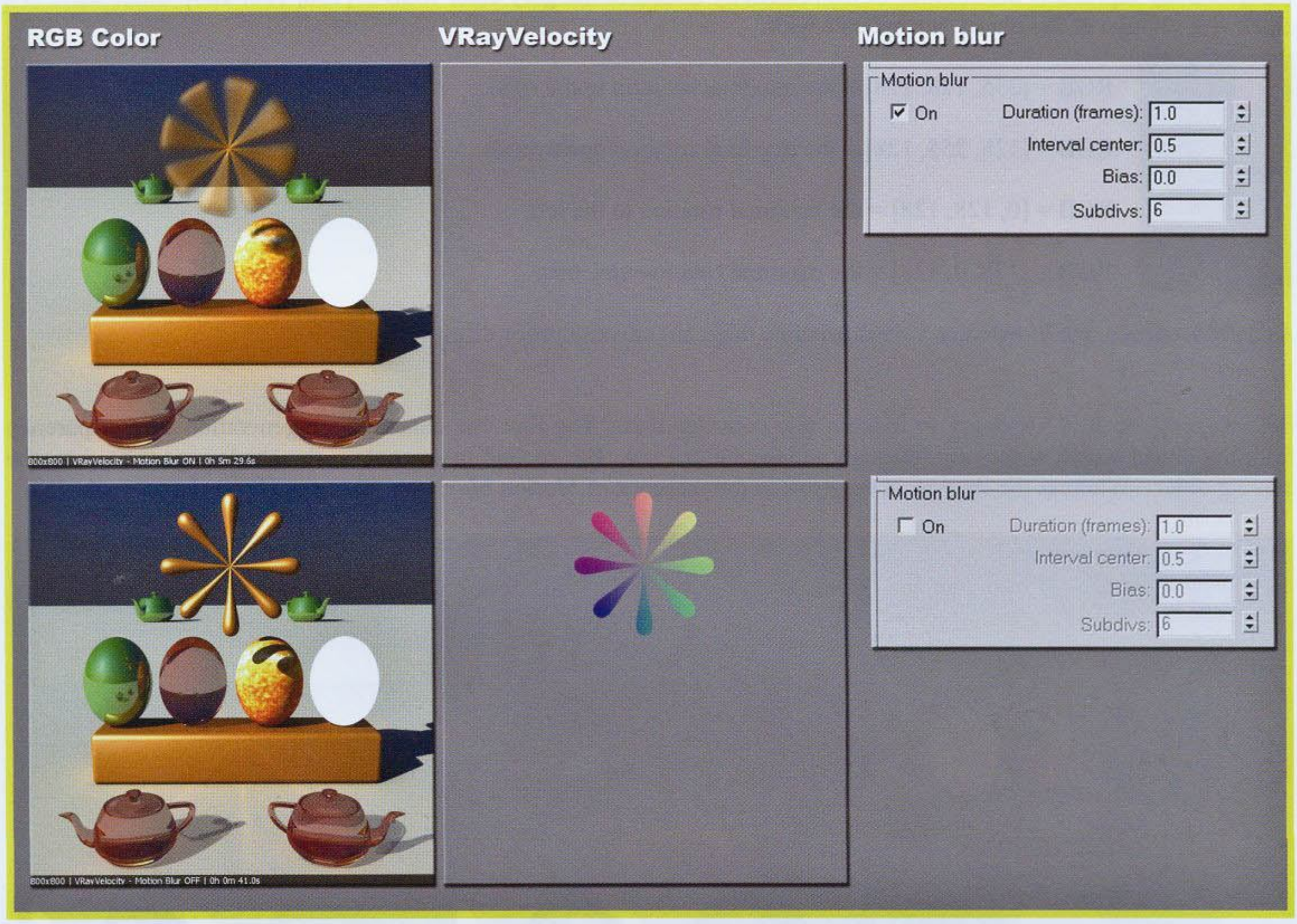
■ **Figure 8.115**
VRayVelocity has five parameters.

The channel has five parameters, of which only one is known to us. Soon the other four are to be examined.

In this scene only the windmill is moving. It is rotating from the left to the right. This animation generates, if it is active, the ***Motion blur*** effect. To control it, it is necessary to change the parameters located in the ***VRay: Camera*** rollout.

■ Figure 8.116
The final render with the ***Motion blur*** effect activated.

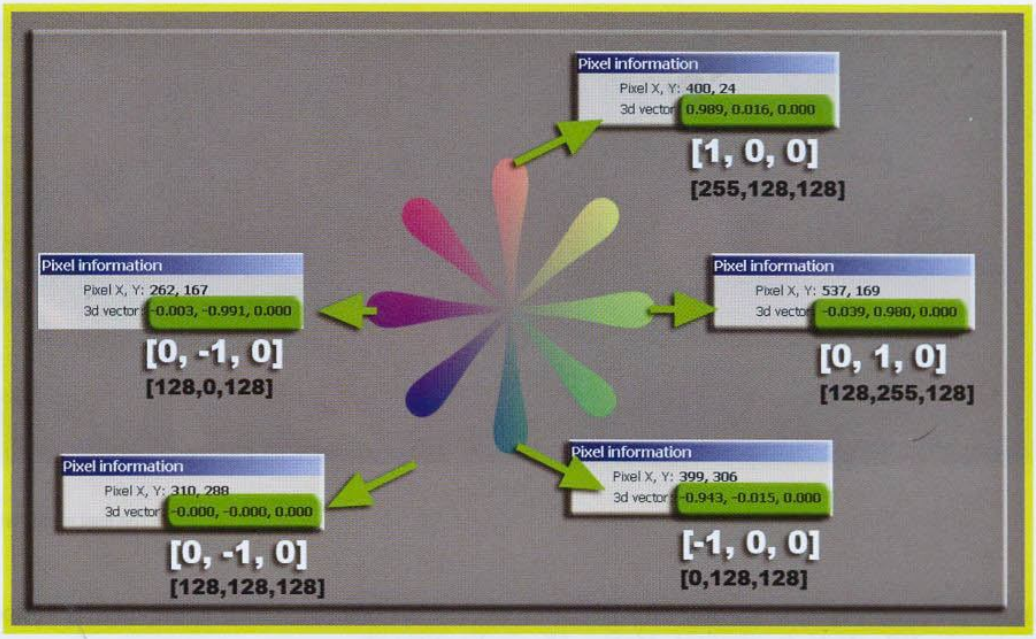
In the previous example ***Motion blur*** is visible since it has been activated in the relative rollout. To visualize this effect within the ***VRayVelocity*** channel, ***Motion blur*** is deactivated.



■ Figure 8.117
Generation of the ***VRayVelocity*** channel.

The ***VRayVelocity*** layer is a channel which defines, through RGB values, the vector direction of the pixels. By analyzing the pixels of the channel with the ***Pixel information*** tool, one can actually observe how the values correspond to a tern of coordinates, with the wording ***3d vector***.

■ **Figure 8.118**
Numerical representation of the vectors. The corresponding colors in RGB are in bold.



VRayVelocity represents the absence of movement by means of the grey color RGB = [128, 128, 128]. Since there is a clockwise rotation of the object, we indicate with:

RGB = [255, 128, 128] = the maximal rotation to the right.

RGB = [128, 255, 128] = the maximal rotation downwards.

RGB = [0, 128, 128] = the maximal rotation to the left.

RGB = [128, 0, 128] = the maximal rotation upwards.

As a consequence, the in-between values generate diagonal movements or displacements.

Before going on, it is necessary to make a few considerations. The first one concerns the generation of transparency from the object which is moving, through which one can see. The second one concerns the shadow generated by the moving object. Whereas the shadow is very soft in the presence of ***Motion blur***, it will be very clear when it is absent.

■ **Figure 8.119**
Transparency and movement, two effects due to the ***Motion blur***.

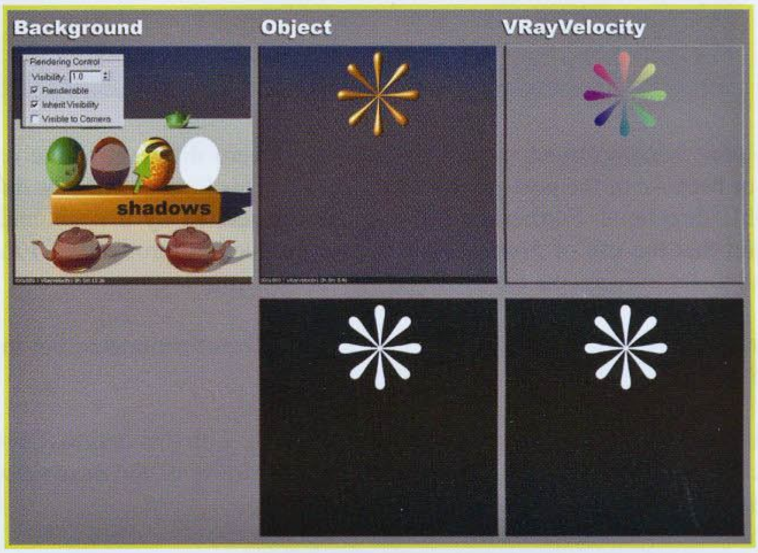
How can one generate an image in ***Motion blur*** in post-production by using a render without ***Motion blur*** and the relative ***Velocity*** channel?

In this case, generating or compositing ***Motion blur*** is not as simple as it is for reflections, refractions, for which a simple mathematical operation between layers is enough.

First of all one has to generate the ***Background*** layer. After one must compute a render containing all the geometries, lights, etc…except for the object which is moving, of which shadows, reflections, refractions, GI, etc.. are to be preserved. This is possible by deactivating the ***Visible to Camera*** option in the ***Object Properties*** panel.

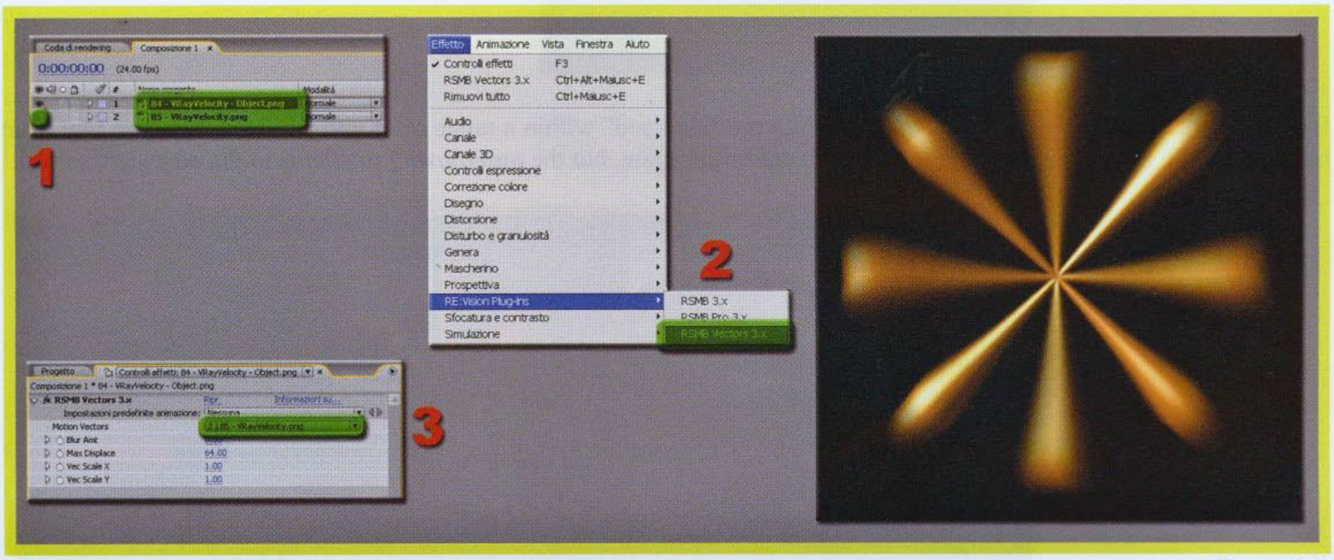
Then one proceeds to create a render containing only the moving object visible in the viewport. All the remaining geometries are hidden. The reason for this is that, together with the render of the object, the correspondent alpha channel is saved.

The last layer is the ***VRayVelocity*** channel, generated by the moving object only. All the remaining geometries are hidden.



■ **Figure 8.120**
The three layers and the relative ***alpha*** channels.

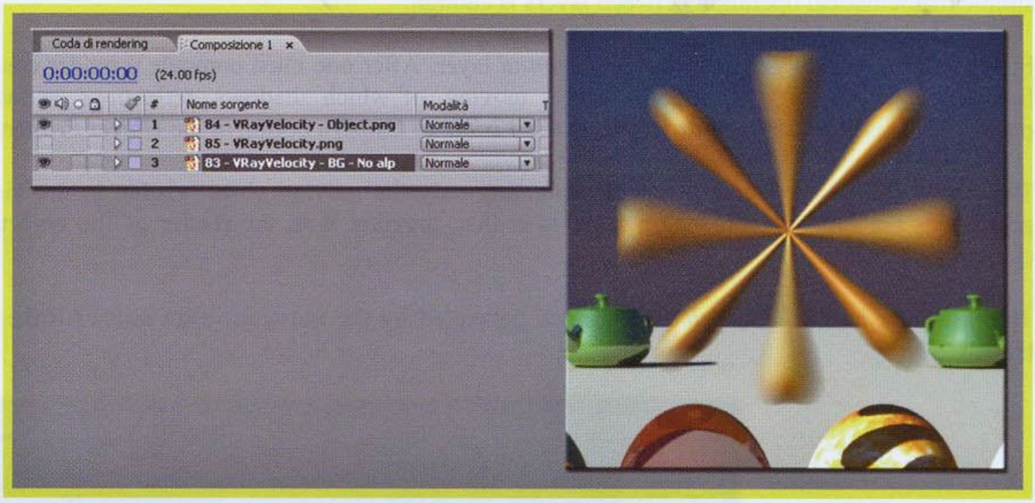
Now a post-production program is necessary, such as Adobe After Effect. Unfortunately, there is no option inside it able to "read" ***VRayVelocity***'s output format. A commercial plug-in exists, which decodes and transforms the information coming from the ***VRayVelocity*** channel in data needed from the ***Motion blur*** application. Its name is ***RealSmart Motion Blur*** and it is available at the address: **http://www.revisionfx.com/products/rsmb/**. Once the two main layers, ***Object*** and ***VRayVelocity***, are inserted into the AE composition, one selects the first one, and applies the ***RSMB Vector*** by using the ***VRayVelocity*** as a layer for the computation of the ***Motion blur***.



■ **Figure 8.121**
Generation of the ***Motion blur*** in post-production.

Now one just has to insert the ***Background*** layer into the composition.

▪ **Figure 8.122**
The final composition.

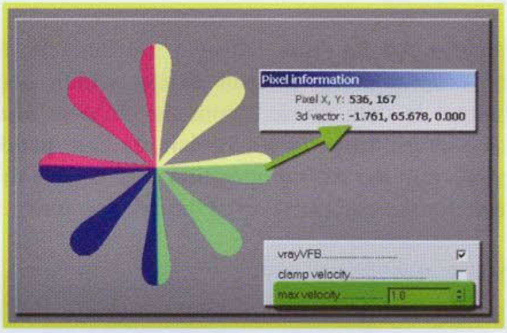


As one can see, ***Motion blur***, accomplished in post-production, is very different to the original one. This does not happen because of ***VRay*** or because of the post-production software. This kind of effect is due to the fact that the object has rotations in all possible directions. Furthermore, the shadow generated by the windmill is not affected at all by ***Motion blur***. It can be said that the use of ***Motion blur*** in post-production through the ***VRayVelocity*** channel can be useful when very short and linear movements are involved.

Clamp velocity - this last option works very similarly to the ***Clamp output*** parameter, but in this case its action affects the ***VRayVelocity*** channel only.

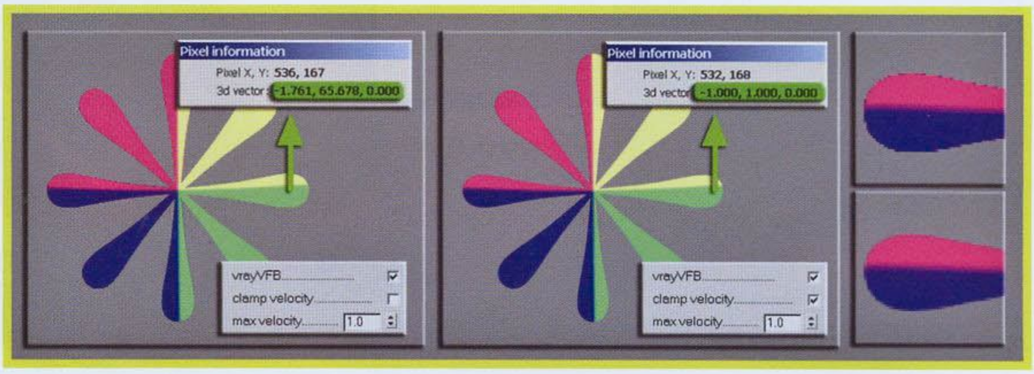
Max velocity - one can set and change the appearance of ***VRayVelocity*** with this option. Observe the following images. The windmill moves clockwise at a very high speed. With very low values of the ***max velocity*** parameter, this makes the ***VRayVelocity*** generate an image with values much higher than 1.0.

▪ **Figure 8.123**
max velocity value set to 1.0. Notice the colors of the ***VRayVelocity*** channel which are too bright.

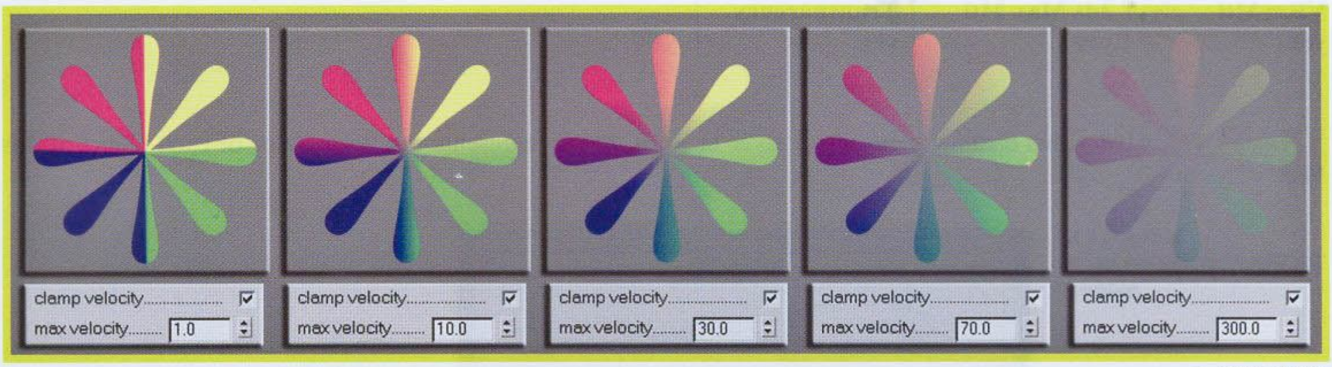


By activating the ***clamp velocity*** option the values are "clamped" within a range of -1 and +1. This way, just like what happens for the ***Clamp output*** parameter, colors are bright again, but the ***antialiasing*** problems of the edges are solved.

▪ **Figure 8.124**
Use of the ***clamp velocity*** parameter. ***Antialiasing*** is correctly computed.

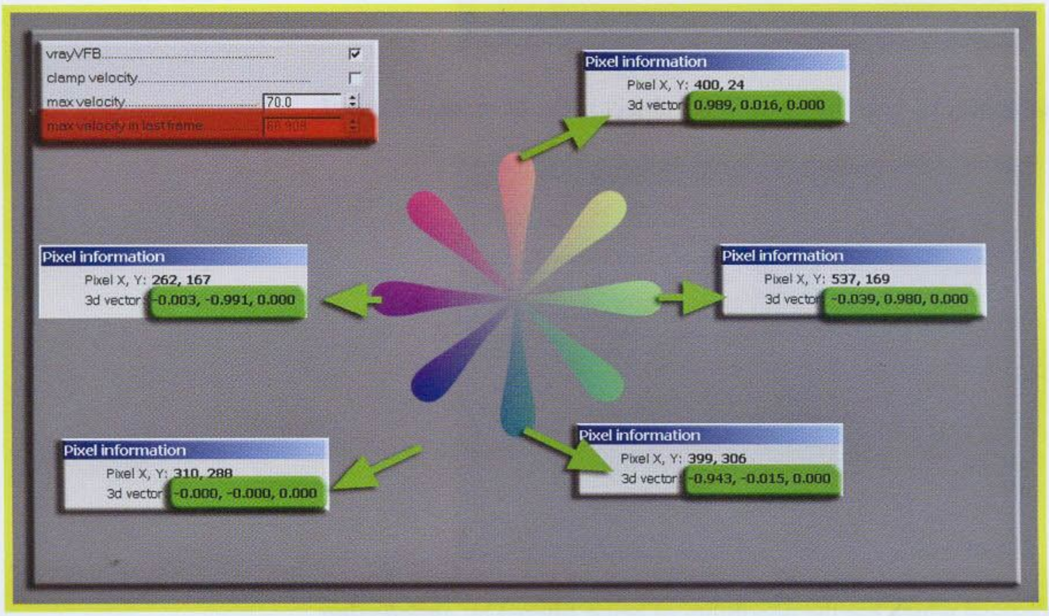


By increasing the ***max velocity*** parameter colors are less saturated, falling within the range -1 and +1.



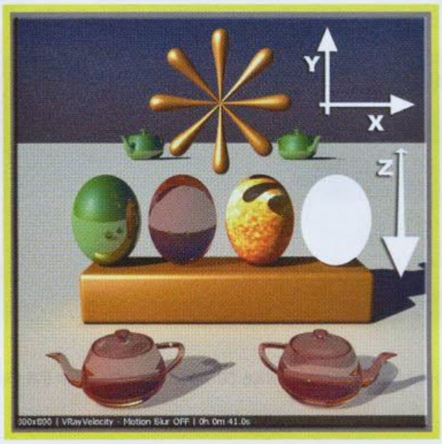
■ **Figure 8.125**
Too high values make the ***VRayVelocity*** channel vanish.

Max velocity in last frame - this parameter, non-modifiable, returns the exact velocity of the frame previous to the one which is being rendered. In the case under examination this value is 66.9. Looking at the previous image one can observe that the channel is devoid of dominants which are too bright and is in the range [-1,1] with a value of ***max velocity*** = 70, with the ***clamp velocity*** parameter deactivated as well.



■ **Figure 8.126**
Settings for a correct generation of the ***VRayVelocity*** channel.

Ignore z direction - by activating this option, the displacements along the z axis, the direction parallel to the observation axis, are not considered. It is active by default, since this translation, in most cases, is negligible for the ***vector Motion blur*** computing. Instead, it can cause problems in the compositing phase. In the exercise, since there is only a rotating movement, the activation or deactivation of this parameter do not lead to any variation in the ***VRayVelocity*** channel.



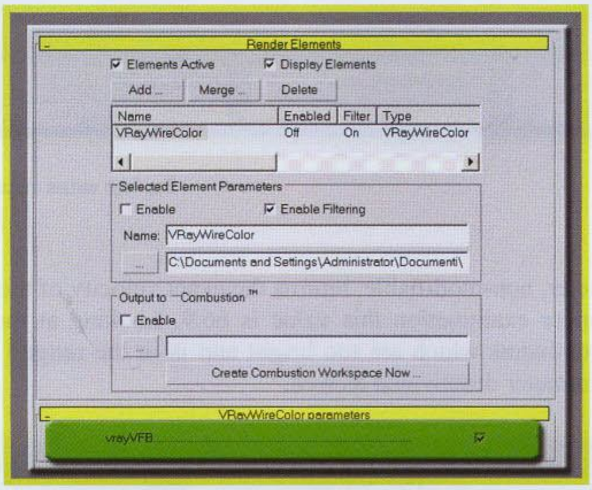
■ **Figure 8.127**
The Z axis coming out of the camera.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayWireColor - this channel generates a layer in RGB which represents the Wire Color of each object of 3ds Max.

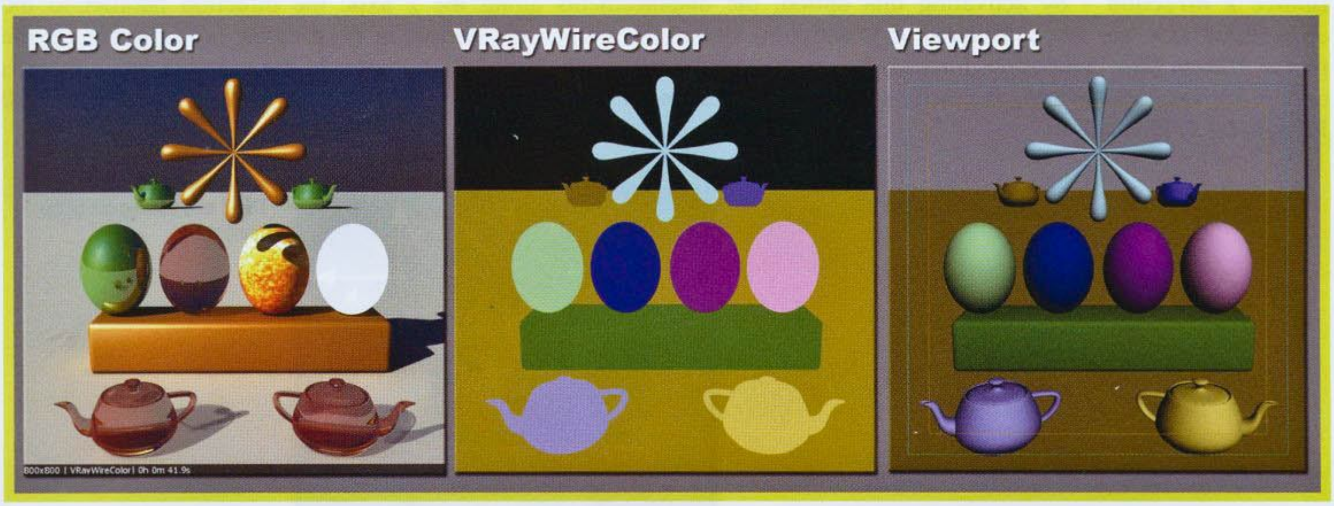
■ **Figure 8.128**
The ***VRayWireColor*** has only one parameter.



The Wire Color is the color appearing near the name of the object in the ***Command Pannel***.

■ **Figure 8.129**
The Wire Color can be visualized directly in the viewport.

The ***VRayWireColor*** is a channel which has the possibility of being filtered by ***antialiasing***.

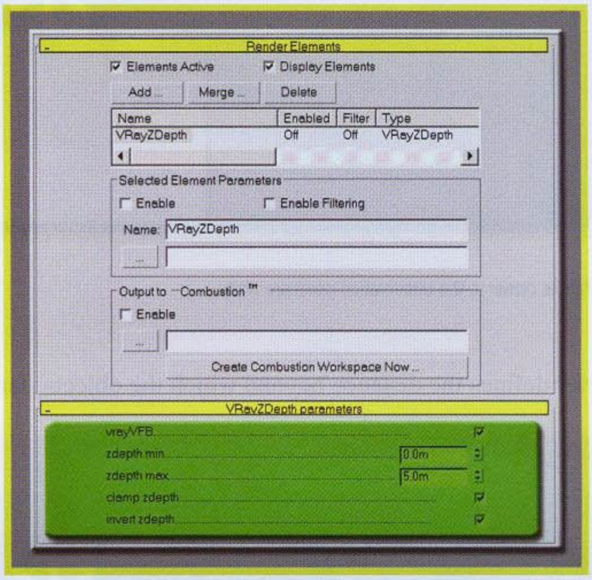


■ **Figure 8.130**
The ***VRayWireColor*** channel. Notice the correspondence between the objects in the viewport and the color of the channel ***VRayWireColor***.

This channel supports:

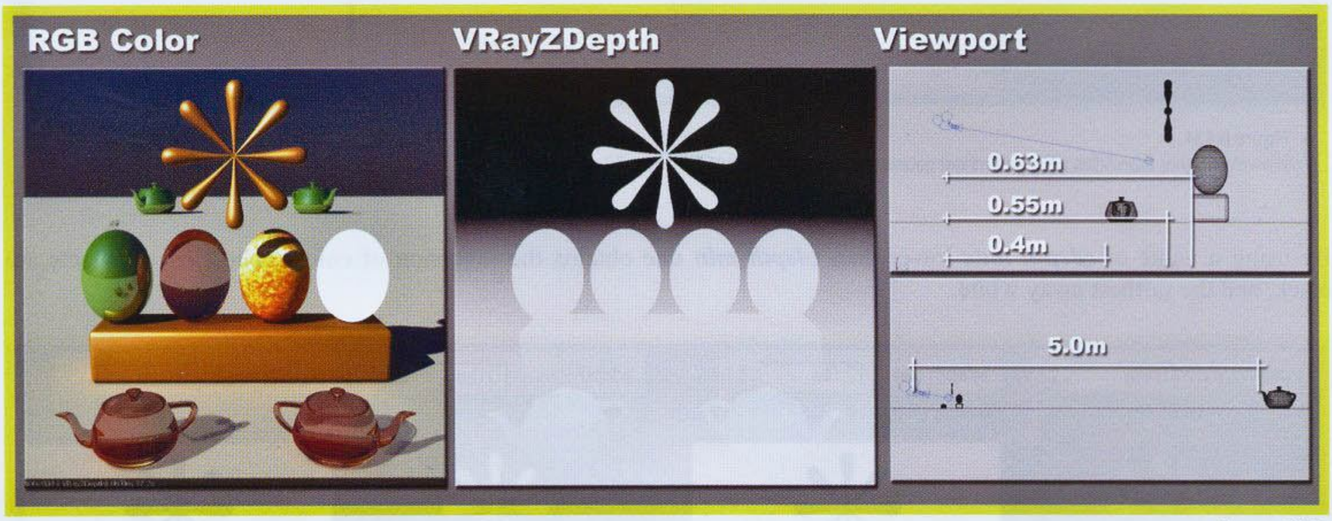
VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

VRayZDepth - this channel allows one to generate a channel in RGB representing the ***ZDepth*** of the image.



■ Figure 8.131
The ***VRayZDepth*** has five parameters, four of which are devoted just to the control of the channel.
Notice the settings of ***zdepth min*** and ***zdepth max*** which have been changed, in this example, respectively to 0.0m and 5.0m.

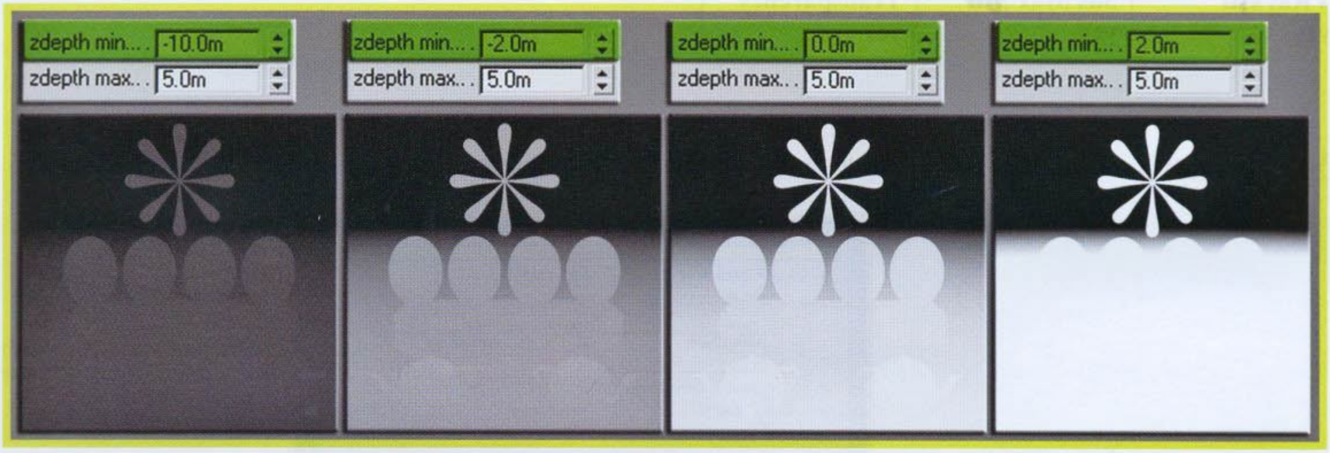
The ***ZDepth*** element is a representation in a grey scale of the Z depth or the depth inside the view, of the objects in the scene. The nearest objects appear white, while the deepest zone of the scene is black. The in-between objects appear grey. The farther an object is from the observation point, the darker the grey will be. The parameter ***zdepth min*** and ***zdepth max*** define the range of depth near and far.



■ Figure 8.132
The ***VRayZDepth*** channel and the relative distances from the point of view. The two farthest tea pots do not appear in the white and black image because they are not positioned over the 5.0m limit.

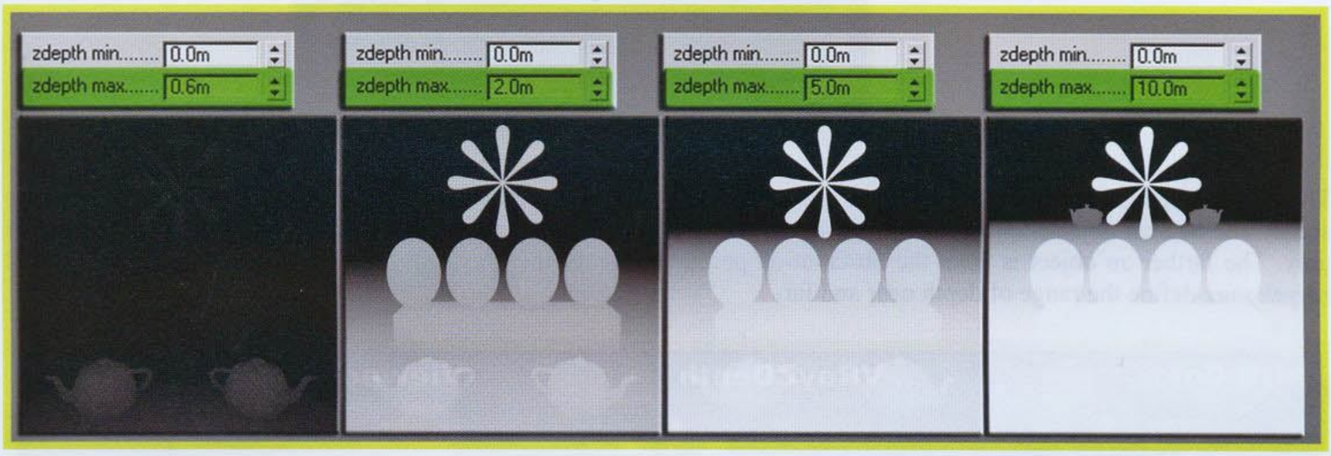
In this example the objects are completely white if positioned at distance 0.0 m from the camera, whereas they are completely black if positioned at a distance of 5.0m. If there are any objects further than 5.0m, it will be necessary to increase the ***zdepth max*** parameter. By increasing the ***zdepth min*** parameter the objects positioned at a distance lower than this value appear completely white.

zdepth min - this parameter defines the distance beyond which the object become visible. The distance may be both positive or negative.



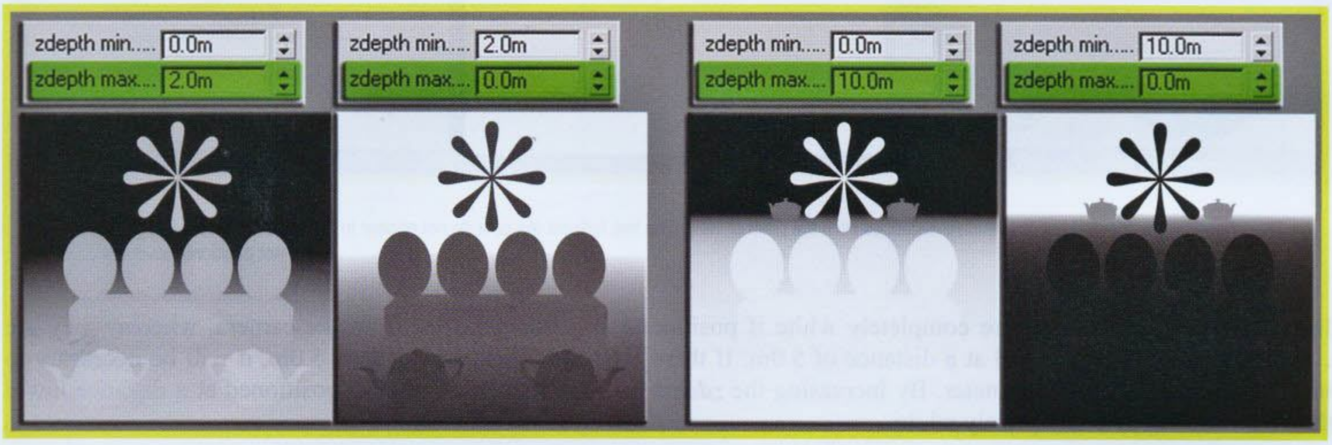
■ **Figure 8.133**
With values higher than 0.0 the objects closer to the observation point are white.

zdepth max - this parameter defines the distance beyond which the objects start to be black.



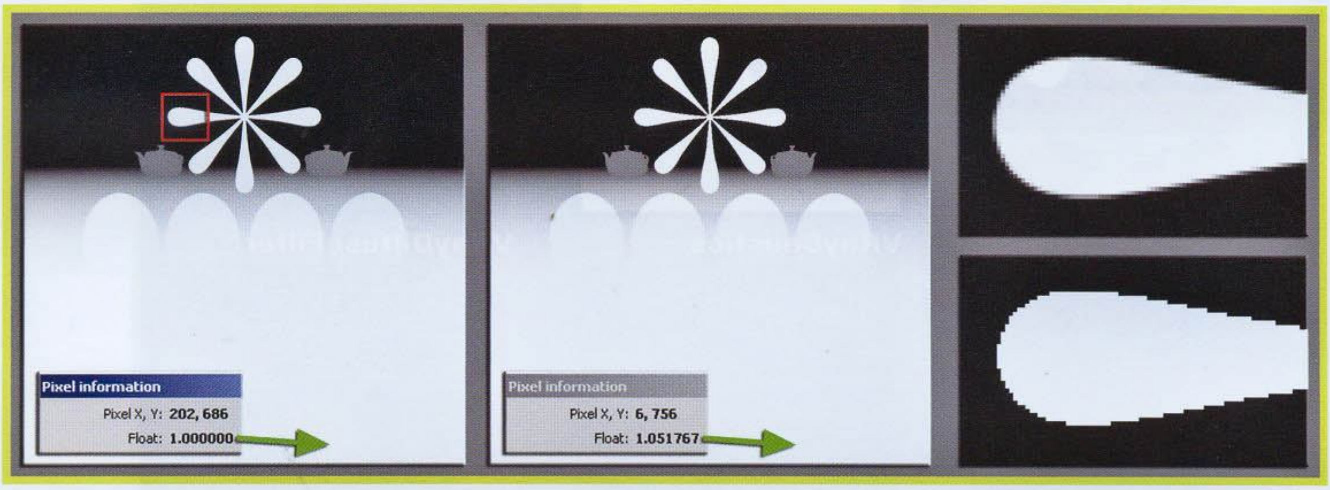
■ **Figure 8.134**
With a value higher than 5.0m also the farthest geometries are visible within the ***VRayZDepth*** channel.

By using a value of ***zdepth max*** lower than ***zdepth min*** one obtains the inversion of colors. The closest objects are black, and the farthest away white.



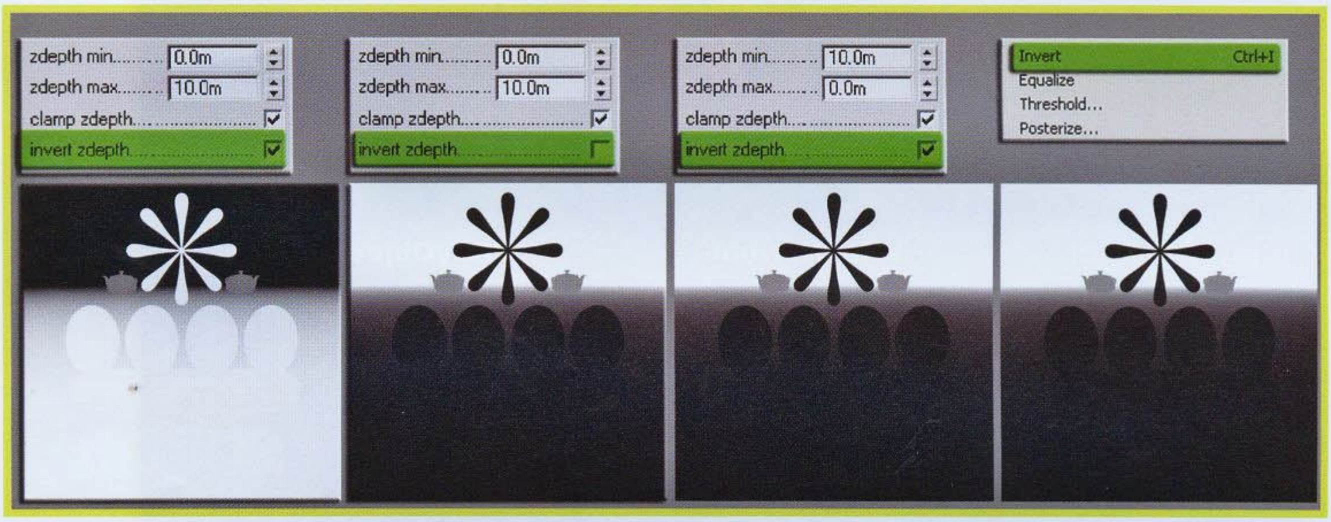
■ **Figure 8.135**
Inverted ***VRayZDepth*** channels.

clamp zdepth - this parameter activates or deactivates the ***clamping*** of the ***VRayZDepth*** channel. By activating it, values higher than 1.0 are normalized to 1.0. By deactivating it, it is possible to generate float number images with values higher than 1.0. But in this case some ***antialiasing*** problems can occur.



■ **Figure 8.136**
Notice the serration due to the deactivation of the ***clamp zdepth*** option.

invert zdepth - this parameter inverts the ***zdepth*** values. What is black becomes white and vice versa.

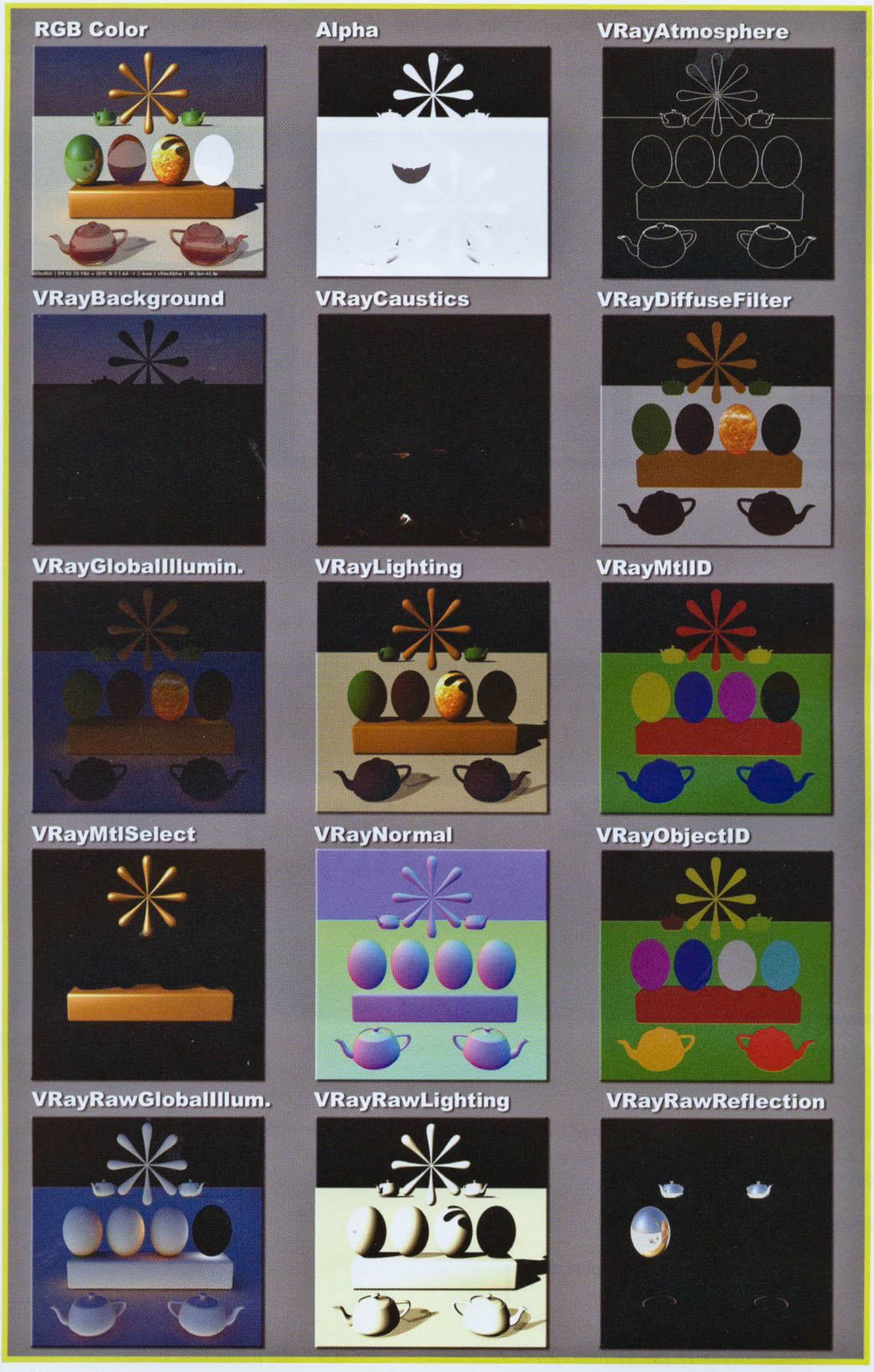


■ **Figure 8.137**
Color inversion of the ***VRayZDepth*** channel.

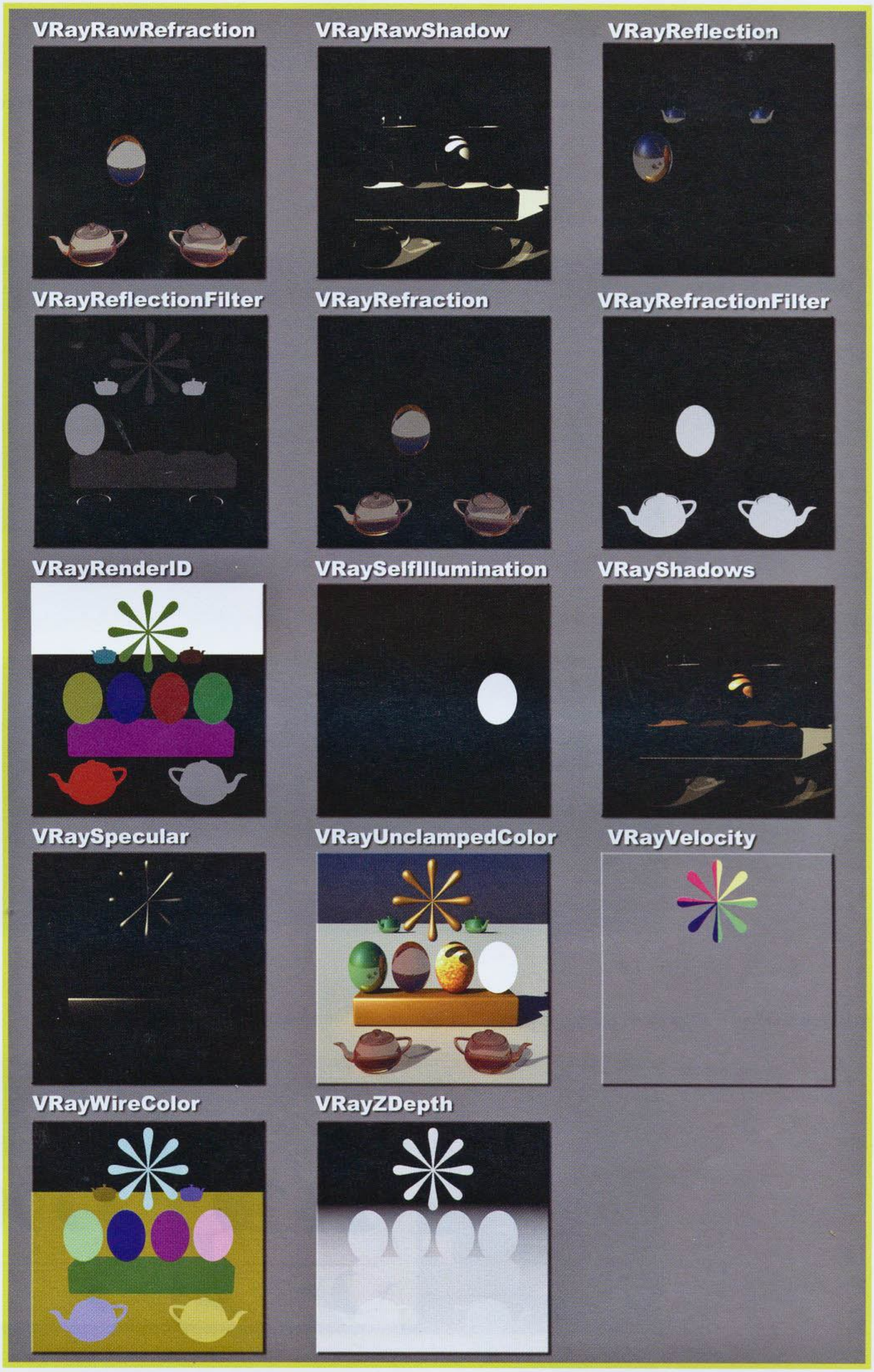
It is possible to obtain the inverse of the ***VRayZDepth*** channel both by using the option ***invert zdepth*** or by inverting the values of ***zdepth min*** and ***zdepth max***, keeping the ***invert zdepth*** parameter active. Another solution is the inversion of the channel with a program like Photoshop.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes



■ **Figure 8.138**
The first 15 ***Render Elements*** of VRay.

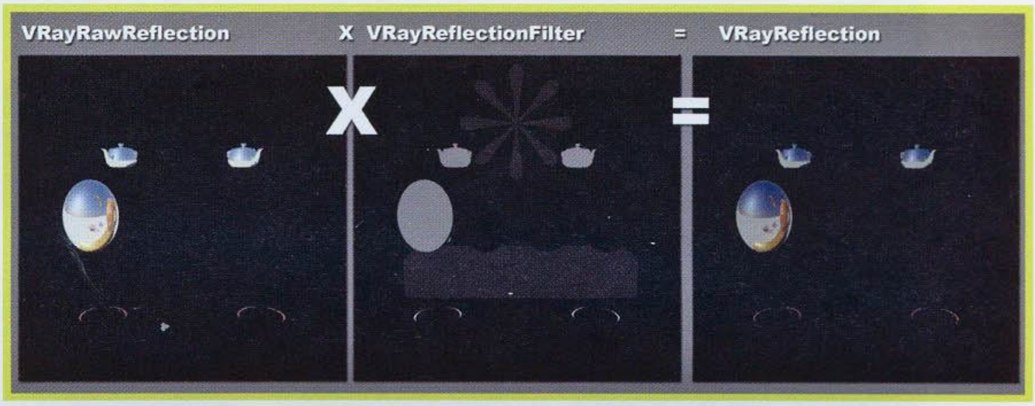


■ **Figure 8.139**
The last 14 ***Render Elements*** of VRay.

COMPOSITING

Before we have observed how two or more layers can be mathematically joined in such a way to fulfil a more complex image. For example, we saw how:

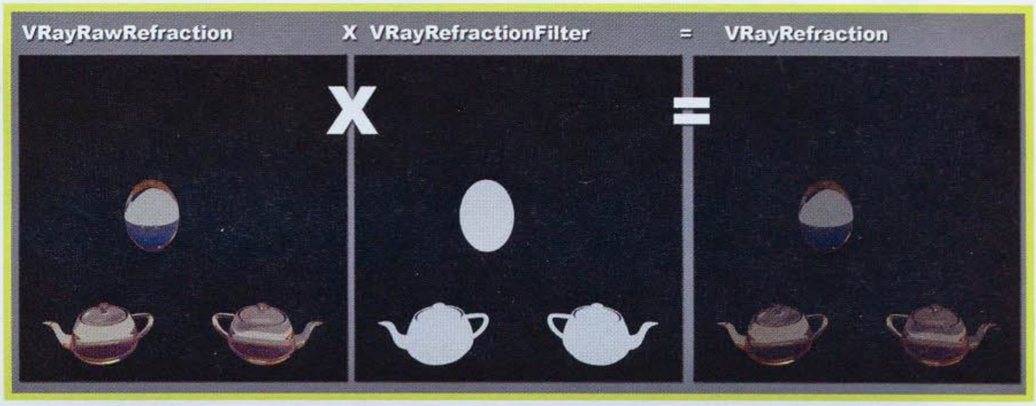
VRayRawReflection** * **VRayReflectionFilter** = **VRayReflection



■ **Figure 8.140**
Multiplication of two channels.

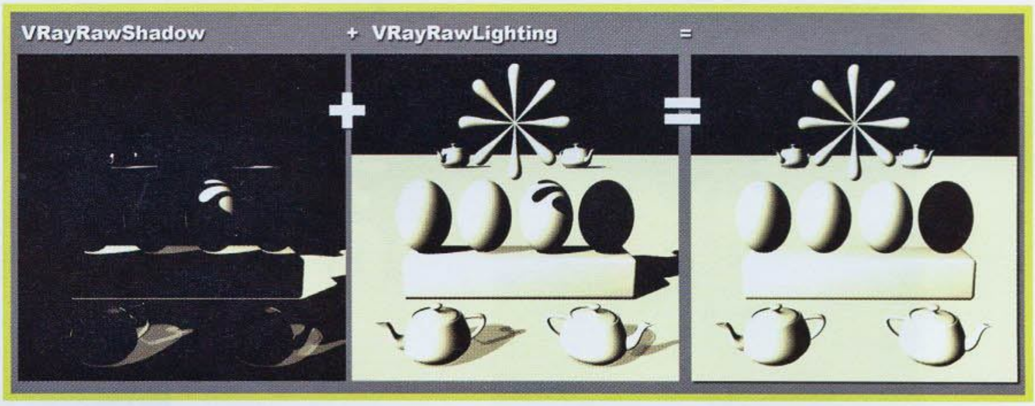
Or:

VRayRawRefraction** * **VRayRefractionFilter** = **VRayRefraction



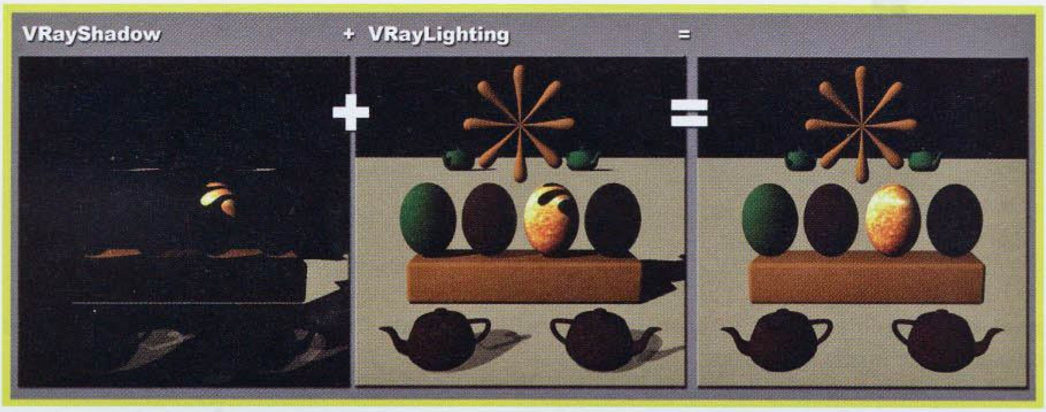
■ **Figure 8.141**
Multiplication of two channels.

VRayRawShadow + ***VRayRawLighting*** = ***VRayRawLighting*** (without shadows).



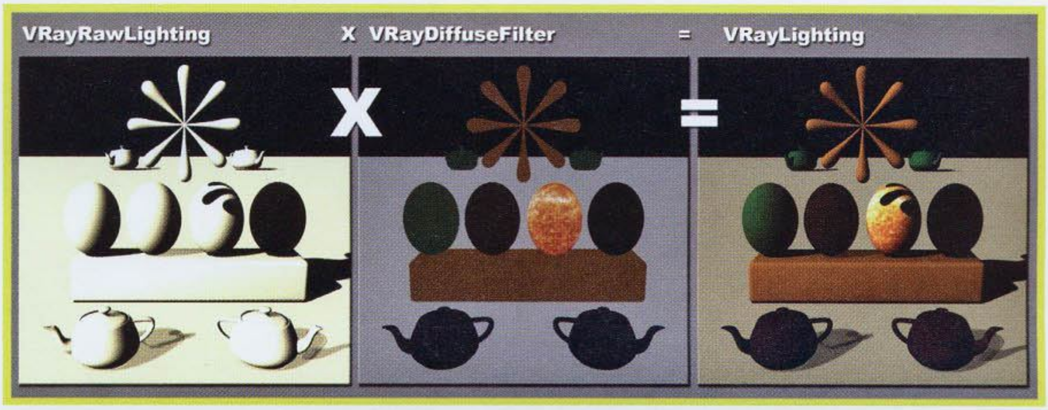
■ **Figure 8.142**
Sum of two channels.

VRayShadow + *VRayLighting* = *VRayLighting* (without shadows).



■ **Figure 8.143**
Sum of two channels.

VRayRawLighting * *VRayDiffuseFilter* = *VRayLighting*



■ **Figure 8.144**
Multiplication of two channels.

VRayRawGlobalIllumination * *VRayDiffuseFilter* = *VRayGlobalIllumination*

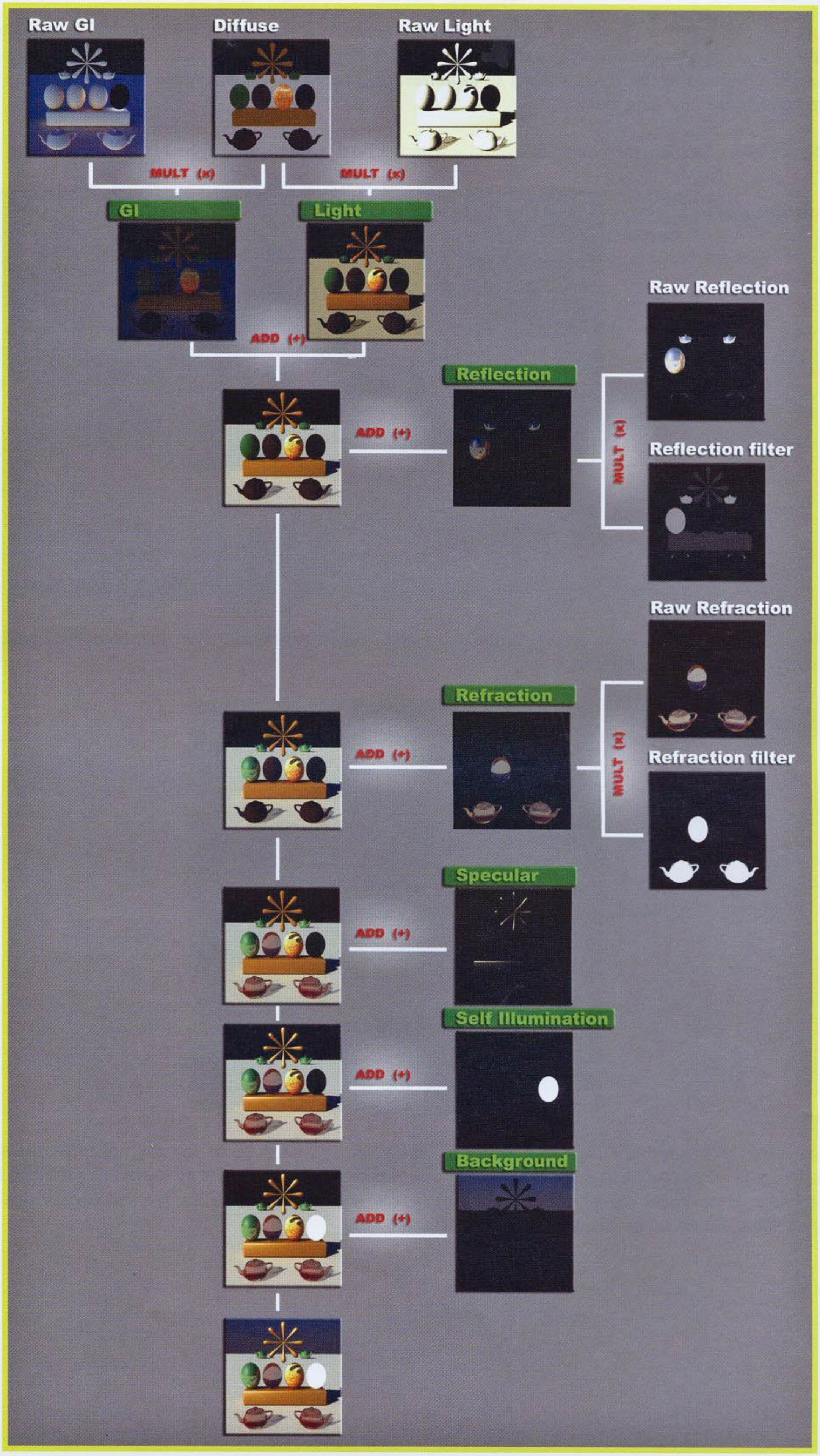
■ **Figure 8.145**
Multiplication of two channels.

These operations are very interesting, but it is possible to go even further, rebuilding the entire render starting from some analyzed channels. The first thing to do is to make sure to be in ***Linear mapping*** mode, undoing any gamma correction. This is necessary to guarantee a correct composition. For the same reason, one has to compute the various channels in float numbers, deactivating any clamping.

In this case one must use:

1. ***Raw GI***
2. ***Diffuse***
3. ***Raw Light***
4. ***Raw Reflect***
5. ***Reflect filter***
6. ***Raw Refract***
7. ***Refract filter***
8. ***Specular***
9. ***Self Illuminaiton***
10. ***Background***

By employing these 10 layers it is possible to mix them in such a way to generate the final render. The next image represents the procedure for linking the layers in just one image.

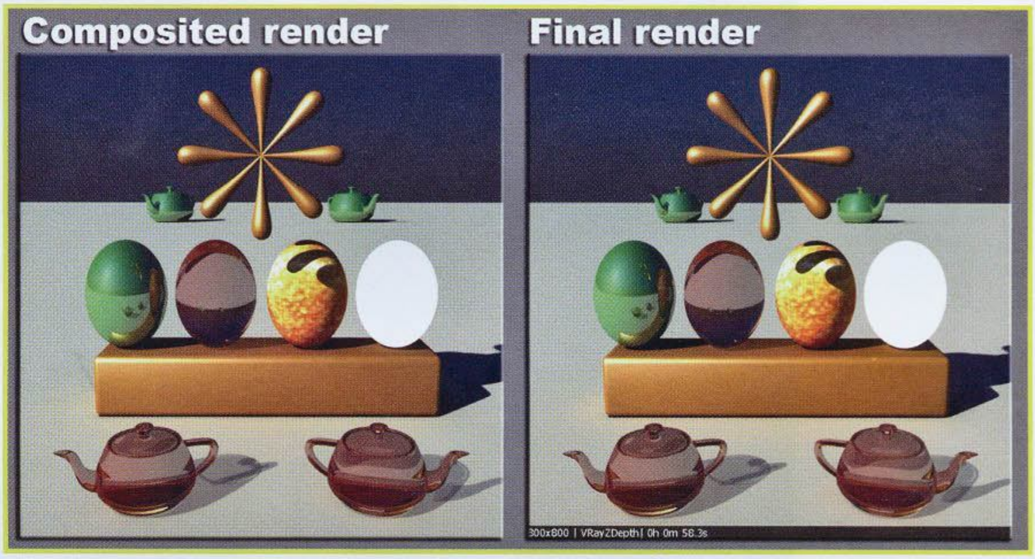


■ Figure 8.146
Scheme of the channels for the generation of the final render.

One can observe how the multiplication operations generate channels that *VRay* is able to create automatically. During compositing it is preferable to accomplish sums of layers only, avoiding multiplications. The reason for this is to avoid having problems in the zones subject to ***antialiasing***. Therefore it is advisable to composite the layers highlighted in green only, by summing them.

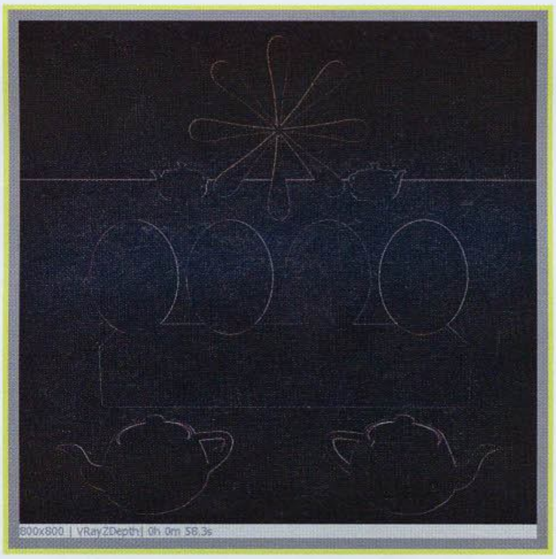
We will now compare the composited render with the final render.

■ Figure 8.147
The two renders are, apparently, the same.



By observing them with the naked eye the two renders might appear the same, but, if they are analyzed with Photoshop, one discovers an interesting feature.

■ Figure 8.148
Difference between the final render and the composited one.



The two renders are not the same in truth. Unfortunately, because of the multiplication of some channels, some differences are appeared in the zones subject to ***antialiasing***. There is no way to fix this problem, because these kind of problems are intrinsic to the multiplication of layers. If during compositing we had carried this out exclusively with layer summations, no problems would have occurred.

VRAY: UPDATES

VRay 1.5 RC3

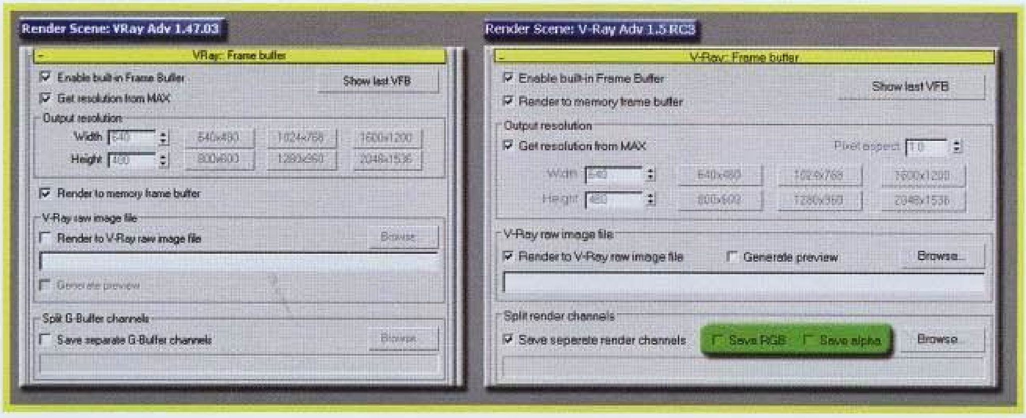
INTRODUCTION

The accomplishment of this book began some time ago, when the latest release of *VRay* was ***v1.47.03***. As the writing of the handbook proceeds, new versions of *VRay* have been released and, each time this has happened, the new version was used for the handbook. This method has lead to the covering of some subjects with non-updated versions. The purpose of this brief chapter is to make up for this loss, by updating the sections with new features introduced in ***VRay v1.5 RC3***.
The main updates treated are the following:

NEW FEATURES	VARIATIONS AND ADDITIONS
VRay: Frame buffer rollout	New options ***Split render channel***.
VRay VFB	***sRGB*** button.
Antialiasing	Rollout separation.
Antialiasing	QMC: ***Use QMC sampler thresh.*** option.
Antialiasing	QMC: ***Clr thresh.***
Antialiasing	QMC: ***Show samples***.
Antialiasing	Adaptive: ***Clr thresh***.
Antialiasing	Adaptive: ***Show samples***.
Antialiasing	New filters: ***VRayLanczosFilter***, ***VRaySincFilter***.
Irradiance Map	Elimination of the ***Blur GI*** parameter.
Irradiance Map	***Detail enhancement***.
Irradiance Map	Modification of the selection of the computing modes.
Light Cache	***Adaptive tracing*** option.
Light Cache	***Use directions only*** option.
rQMC	Path sampler menu: ***Default*** and ***Latin super cube***.
Color mapping	Corrected names for the ***Tone mapping*** options.
Color mapping	New ***Gamma*** option.
Color mapping	***Gamma*** and ***Reinhard option***.

PARAMETERS

Frame buffer rollout



■ **Figure 9.1**
New saving options of the channels.

Save RGB - this parameter allows one to save the ***RGB*** channel if the ***Save separate render channel*** option is selected. For details see the chapter concerning compositing.

Save alpha - this parameter allows one to save the ***alpha*** channel if the ***Save separate render*** channel option. For details see the chapter concerning compositing.

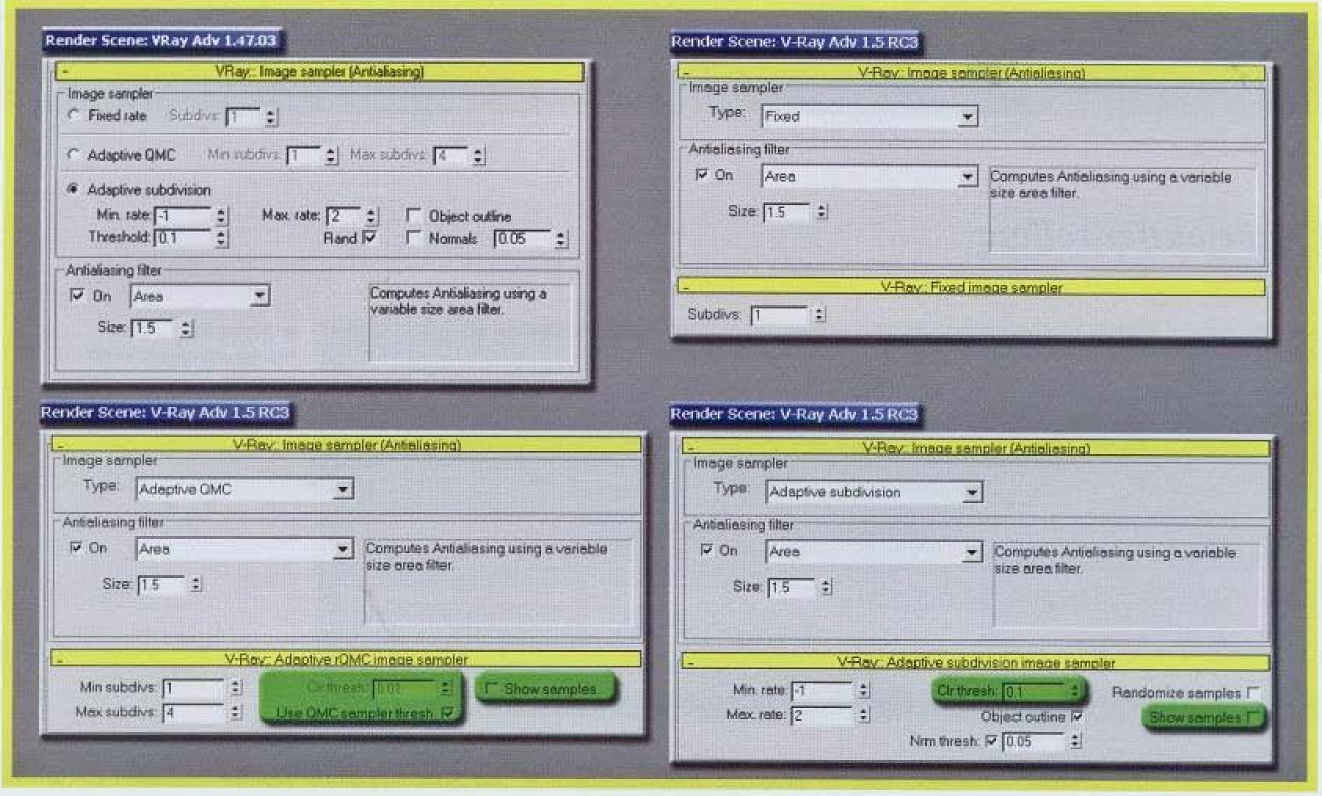
VFB

■ **Figure 9.2**
The ***sRGB*** function.

sRGB - this button allows one to visualize the image in the ***sRGB*** color space within the ***VFB***. For details, it is advisable to read the chapter concerning the rollout ***VRay: Color mapping***.

Image sampler (Antialiasing)

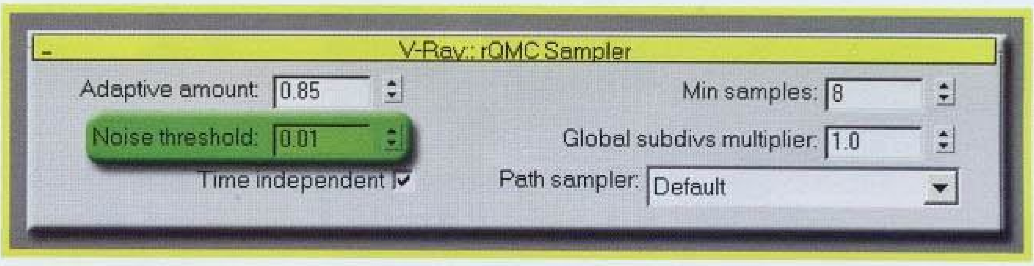
The ***antialiasing*** options have been divided into two different rollouts in the version 1.5 RC3 of ***VRay***. The first contains the pull-down menu with which it is possible to select both the ***antialiasing*** computing mode and the type of filter. The second rollout, which changes according to the mode employed, returns the options respectively of the methods ***Fixed***, ***Adaptive QMC*** and ***Adaptive subdivision***, selectable in the first rollout.



■ **Figure 9.3**
The new subdivision of the menus for the ***antialiasing*** and the additional parameters.

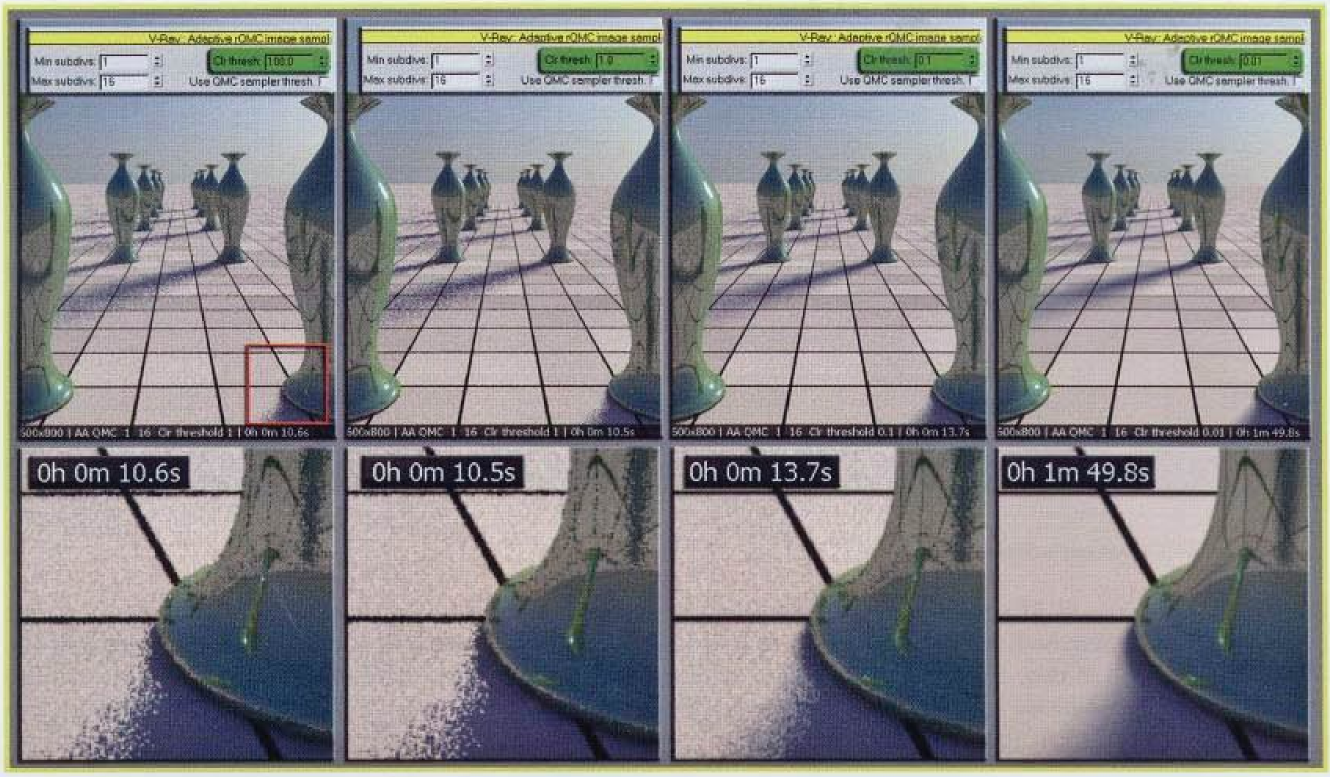
Adaptive rQMC (Antialiasing)

Use QMC sampler thresh - the ***QMC*** method uses an additive sampling system. When ***VRay*** finds pixels for which ***antialiasing*** is necessary, it begins to sample them just once (***Min subdivs*** = 1), increasing the sampling till it reaches the set subdivision (***Max subdivs*** = 4). But how can it know that a new subdivision in pixels is necessary in that specific point? This information is present inside the ***Noise threshold*** parameter, parameter of the ***VRay: rQMC Sampler***.



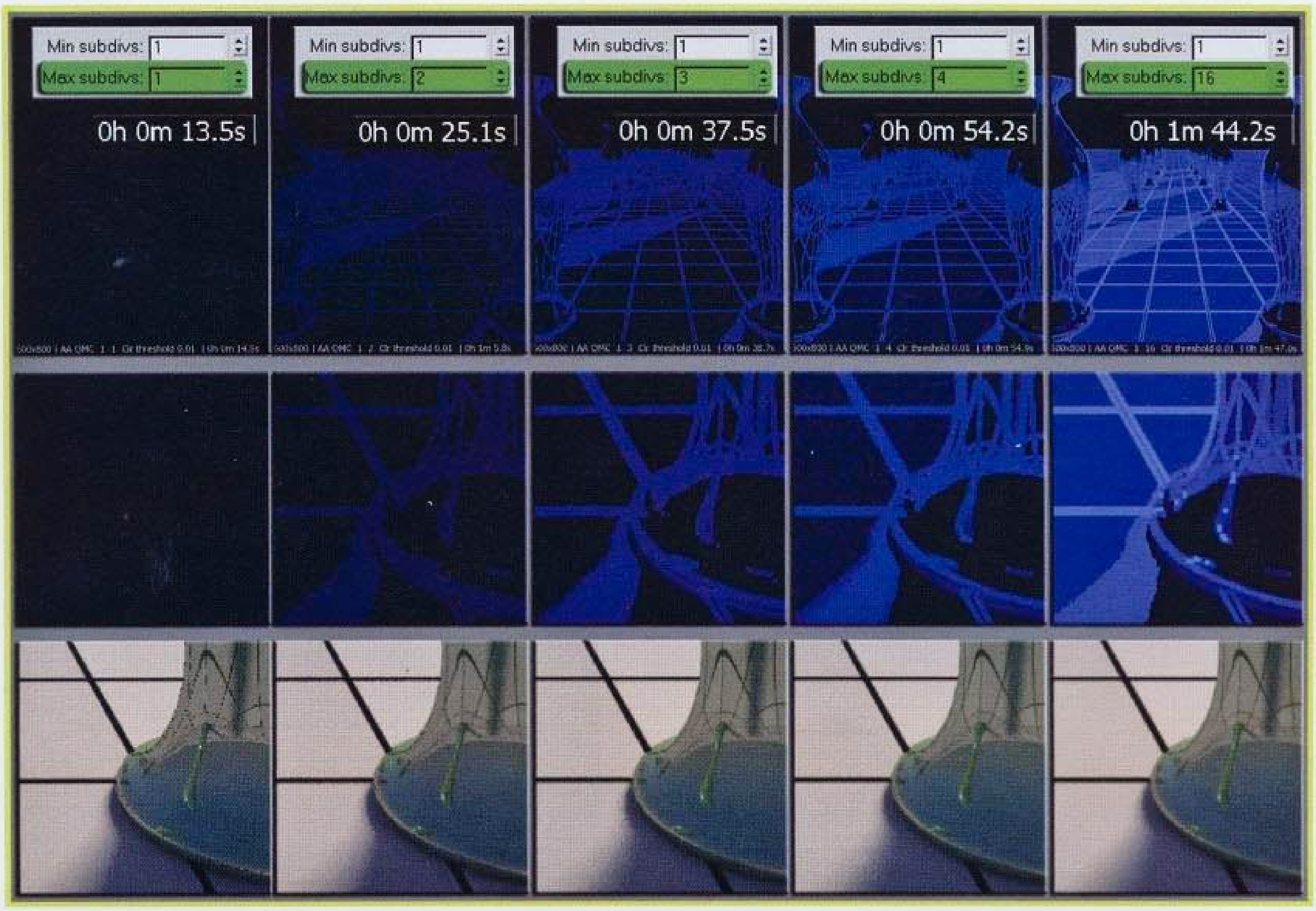
■ **Figure 9.4**
The ***Noise threshold*** parameter is responsible for the quality of all the blurry effects. It defines, for a specific effect, the threshold within which another quality jump is necessary.

It defines the threshold above which it is necessary to subdivide the pixels even more. By default, for the ***antialiasing Adaptive QMC***, it is the general ***Noise threshold*** parameter which establishes whether a pixel has to be subdivided or not. By deactivating this parameter it is possible to choose a different threshold value for ***antialiasing*** relative to the general one set by the ***Noise threshold*** parameter. By lowering this value, the quality of ***antialiasing*** improves at the expense of the time. It is advisable not to go under the value 0.005. If this should happen one would risk having very long rendering times.



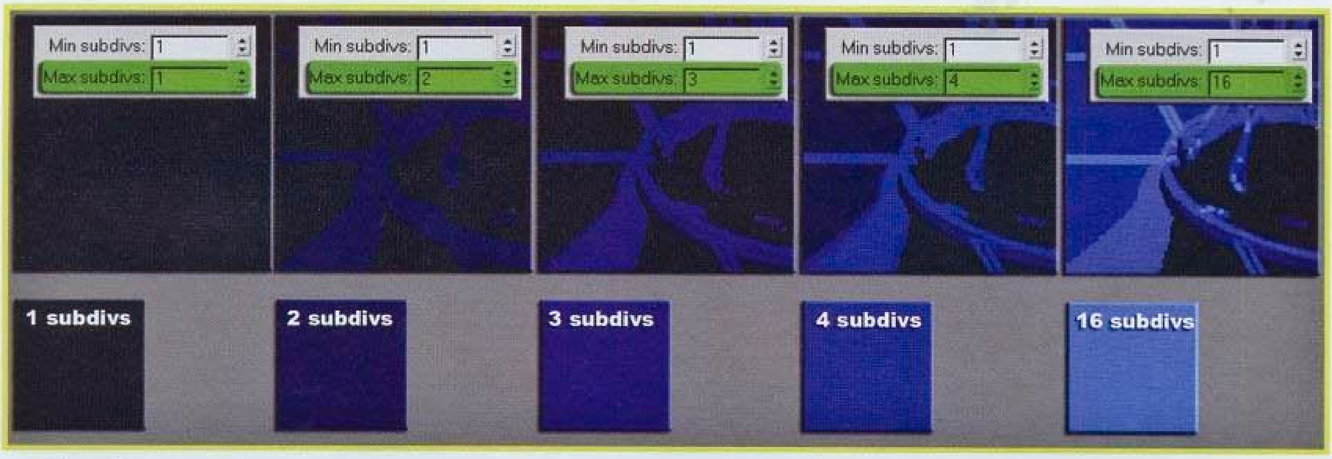
■ **Figure 9.5**
The ***Clr thresh*** option controls the quality of the ***antialiasing***.

Show samples - by using this option an image representing the ***antialiasing*** sampling by means of blue hues is shown instead of the final render. The brighter the pixels are, the higher the sampling (subdivisions) is.



■ **Figure 9.6**
Graphic representation of the samples used for the ***antialiasing***.

Each color represents a different number of ***subdivisions***.



■ **Figure 9.7**
Different samplings correspond to different colors.

By using equal values of ***Min subdivs*** and ***Max subdivs***, the image is plain-colored, because just one value exists, like for the first case, where both the values were set to 1. Bringing the ***Max subdivs*** = 2, some pixels might need just one subdivision, and others two, depending on the value of ***Nrm thresh*** or ***Clr thresh***: therefore pixels are brighter. In this case the image is composed by two colors, one representing the pixels with one subdivision, and the ones with two subdivisions.

Adaptive subdivision (Antialiasing)

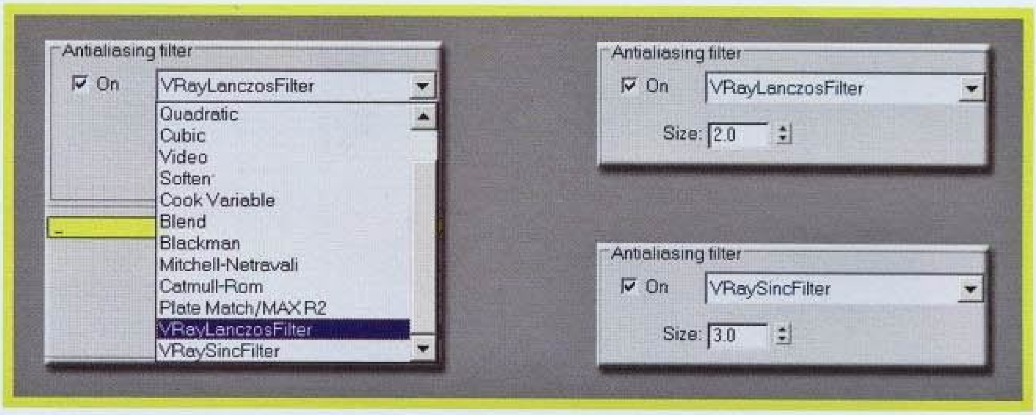
Clr thresh - by using the ***Adaptive subdivision*** method, instead of what happens for the ***Adaptive QMC*** method, the noise threshold is always available separately from the parameters present in the ***VRay: rQMC rollout***. This can be noticed from the default value of 0.1, it is 10 times greater than the same value in the ***Adaptive QMC***.

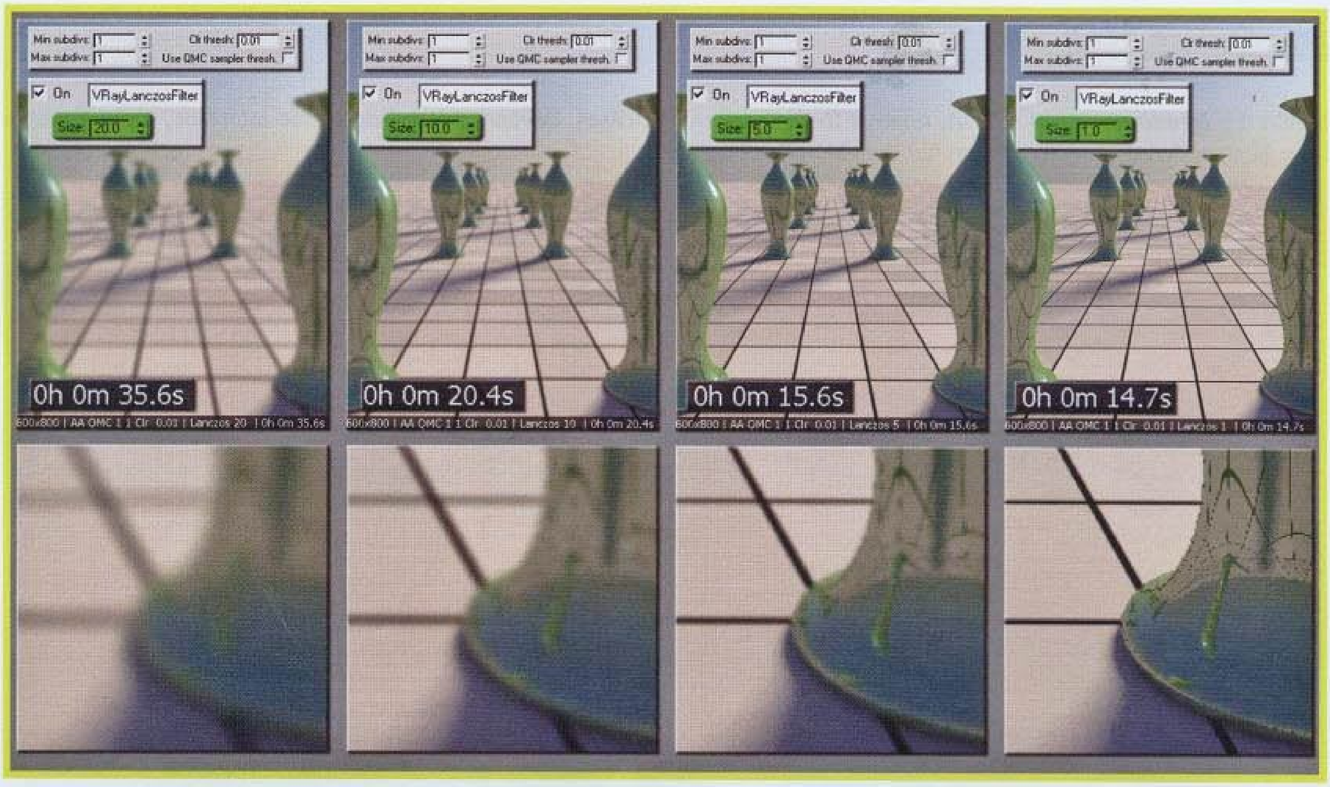
Show samples - by activating this parameter, an image representing the sampling of ***antialiasing*** by means of blue shades appears instead of the final render. The brighter the pixels are and the higher the sampling (the subdivisions) is.

ANTIALIASING FILTER

In the last version of *VRay* two new filters have been introduced: the ***VRayLanczos*** and the ***VRaySincFilter***. The first is a "smooth"-type filter, while the second is a "sharp"-type one. There are no particular advantages in using one rather than the other, even though it is advisable to render with smooth-type filters cleaning up the image in post-production, rather than using sharp filters directly in *VRay*, since *VRay* works using float numbers.

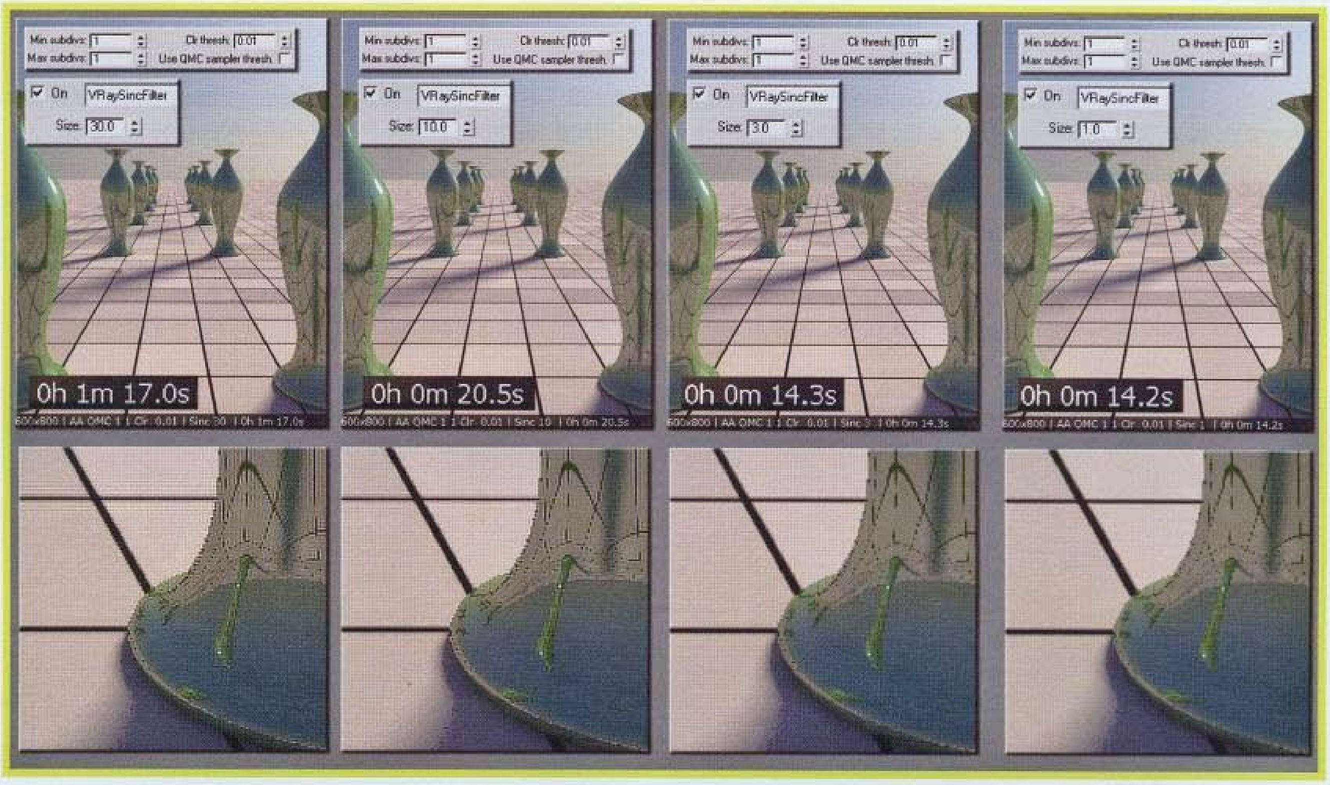
■ **Figure 9.8**
Each filter of the two new ones has just one control parameter.





■ **Figure 9.9**
The ***VRayLanczosFilter*** is, in some respects, similar to the Area one.
--- DVD ---

The higher the value of the ***VRayLanczosfilter*** is the more the image is blurred, with rendering times which get higher and higher.

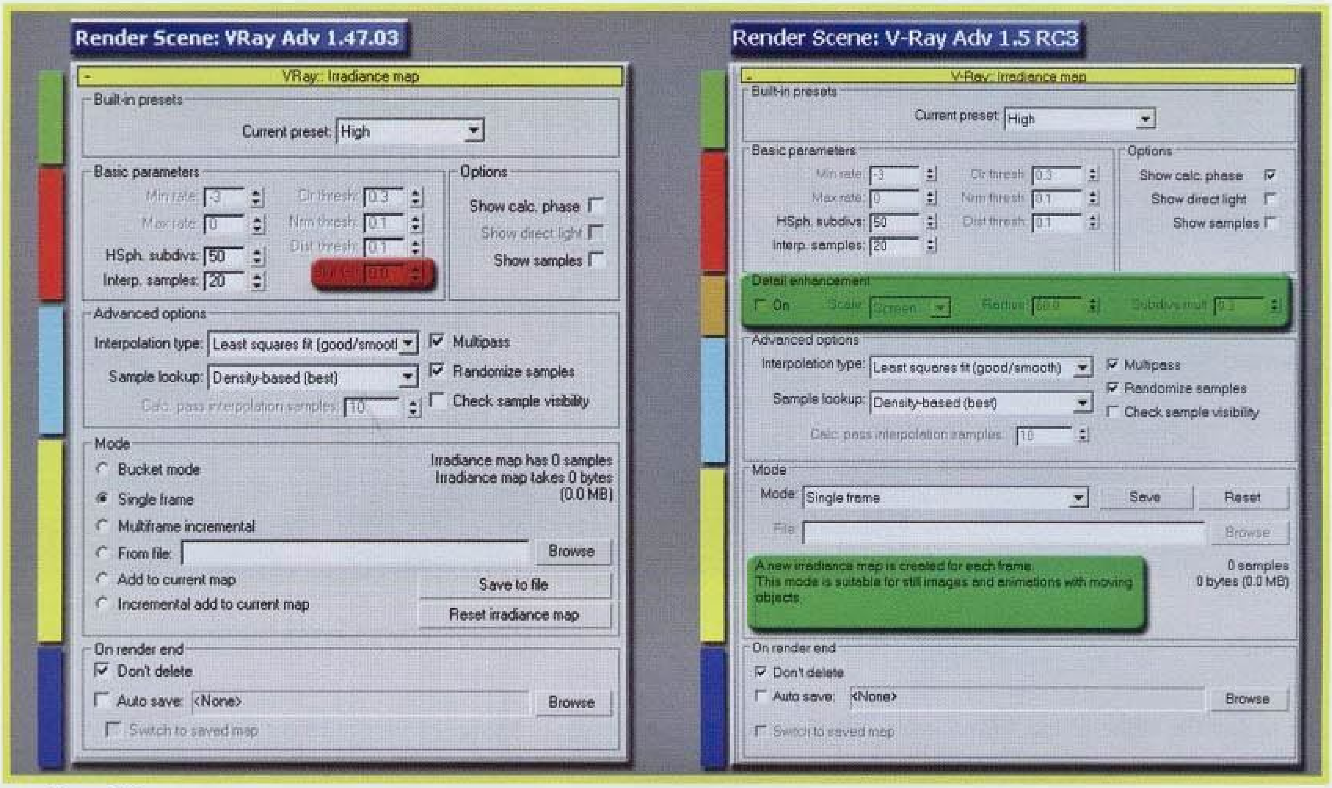


■ **Figure 9.10**
The ***VRaySincFilter*** generates some particular "undulations" around the edges for increasing the contrast.
--- DVD ---

By using the ***VRaySincFilter*** the render is much cleaner and brighter. The higher its value is, the cleaner the image. But one has to pay attention not to use values which are too high, because some problems can occur, like doubling of edges or artefacts.

Irradiance Map

Inside the ***Irradiance Map*** rollout, relative to the previous releases, beyond a restyling of the interface, some features have been added or removed when not considered necessary.



■ **Figure 9.11**
The changes carried out on the Irradiance Map since the v1.5 RC3 of VRay.

Blur GI - the ***Blur GI*** parameter, colored in red, has been deleted. Instead of it, new parameters have been added in order to reduce the flickering during the animations. They are grouped together in the ***Detail enhancement*** section.

Detail enhancement - the aim of these parameters is to help the ***Irradiance Map*** in the computation of the GI in the zones where the Global Illumination is more difficult to compute, like angles, small details or complicated meshes. The options of the ***Detail enhancement*** section are useful especially in animations, where the employment of the ***IM*** can generate flickering problems. To better understand the idea, a very simple scene is rendered. It is an abstract mesh, made up of both homogeneous and flat parts, and geometrically more complicated zones. Everything is illuminated by the ***VRaySky*** only.

■ **Figure 9.12**
The scene has uniform zones, such as the vertical walls, and more complex geometries as well.

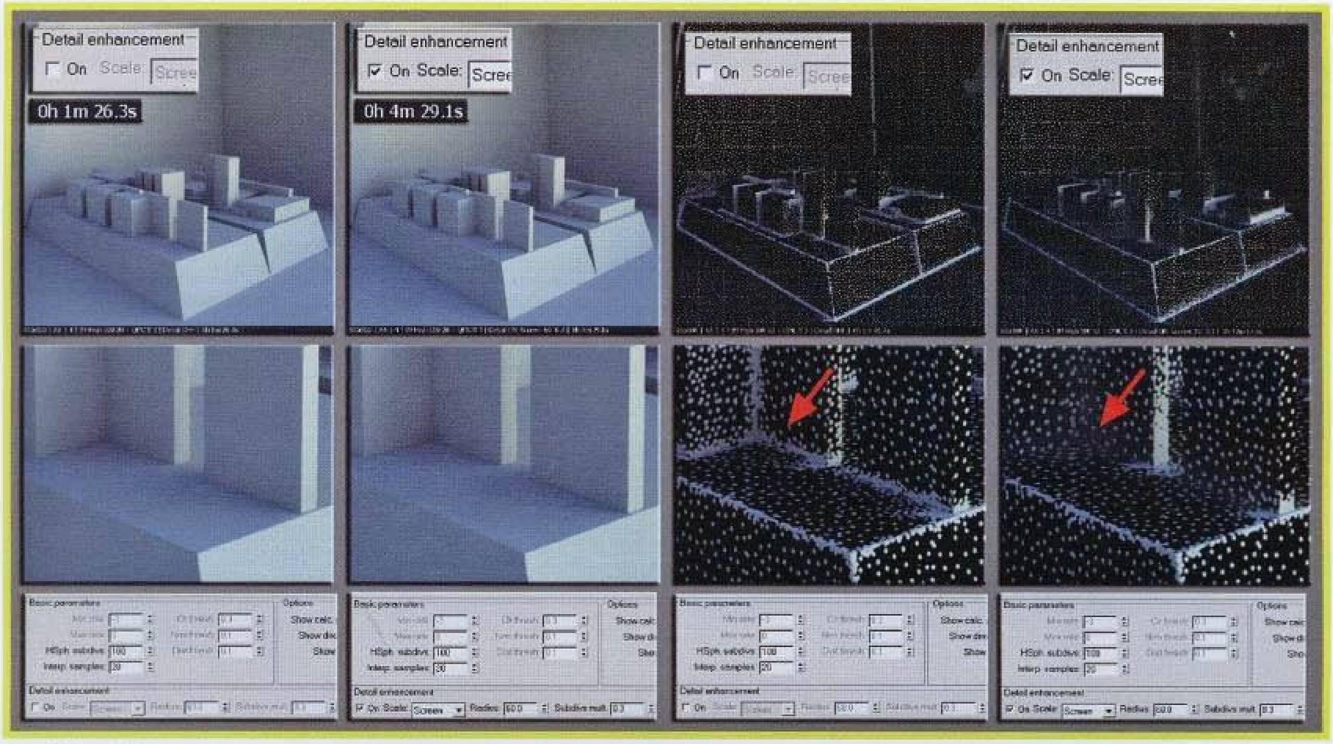
By rendering the scene at 800x800 pixels of resolution, with low values of ***IM*** and ***QMC***, in 14 seconds it is possible to generate the first image at medium quality. Some problems are present, such as blotchy zones and "flying" objects. An increase in the quality of the ***IM*** might be a possible way of solving them.

■ **Figure 9.13**
Comparison between two different qualities of the ***IM***.

By increasing the ***IM*** parameters, the quality of the GI increases exponentially, together with the rendering time. In fact, the time has gone from 14 sec. to 1 min. and 30 sec. Paying attention to the images with the samples, one can notice how ***VRay*** positions the ***IM*** samples where more complicated zones are present, since a higher precision is necessary exactly in those points. And exactly in those zones, where a higher density of samples is present, the ***Detail enhancement*** shows its usefulness. By activating it, ***VRay*** is uses the ***QMC*** method instead of the ***IM*** in those zones.
VRay goes on computing the ***IM*** as always, but in zones with a higher concentration of samples, it passes from the ***IM*** to the ***QMC*** method for ***GI*** computation. In this case the computing method is ***IM+QMC***. Therefore, in flat and undetailed zones, ***VRay*** computes the ***GI*** by using the ***IM+QMC*** method. But in the corners the ***QMC+QMC*** method is used. Obviously, this leads to an increase in rendering time, but also returns a higher quality, deleting possible problems when animations are present.
What happens in the case of ***Fly-through*** animations? The ***GI*** is simply computed each ***n*** frames through the ***IM+QMC*** or the ***IM+LC*** system, without using ***Detail enhancement*** (it would be a waste of time). Afterwards the ***IM*** and/or the ***LC*** is loaded in the memory, and the ***Detail enhancement*** options are activated. This way one ensures that there will be no problems during animations. Certainly, by using this system, the GI is computed twice in critical areas. Firstly with the ***IM*** every "***n***" frames. And after for each single frame, through the ***Detail enhancement***. This is not that bad, it is the price to pay for having a higher quality. Another method could be the use of the ***Detail enhancement*** with very low values during the preparation of maps.

Many may have realized how this approach to the computation of the GI is very similar to the technique used by the ***VRayDirt*** map. ***VRay*** uses the same occlusion method in deciding where to compute the GI via the ***Detail enhancement*** system. Where closeness between the geometries is higher, the GI is computed with the ***QMC***. Otherwise, one continues to use the ***IM***. The radius parameter is used for deciding the amplitude of the zones to treat with the ***Detail enhancement***.

In the following image the option ***Detail enhancement*** is active and has high ***IM*** values (***IM High 100 20***), providing both the final version and the one with visible samples. Nothing else has been modified. It is possible to notice how the samples which were arranged along the corners when only the IM was used, have now disappeared. Instead of them the ***GI*** is computed through the ***QMC*** method.

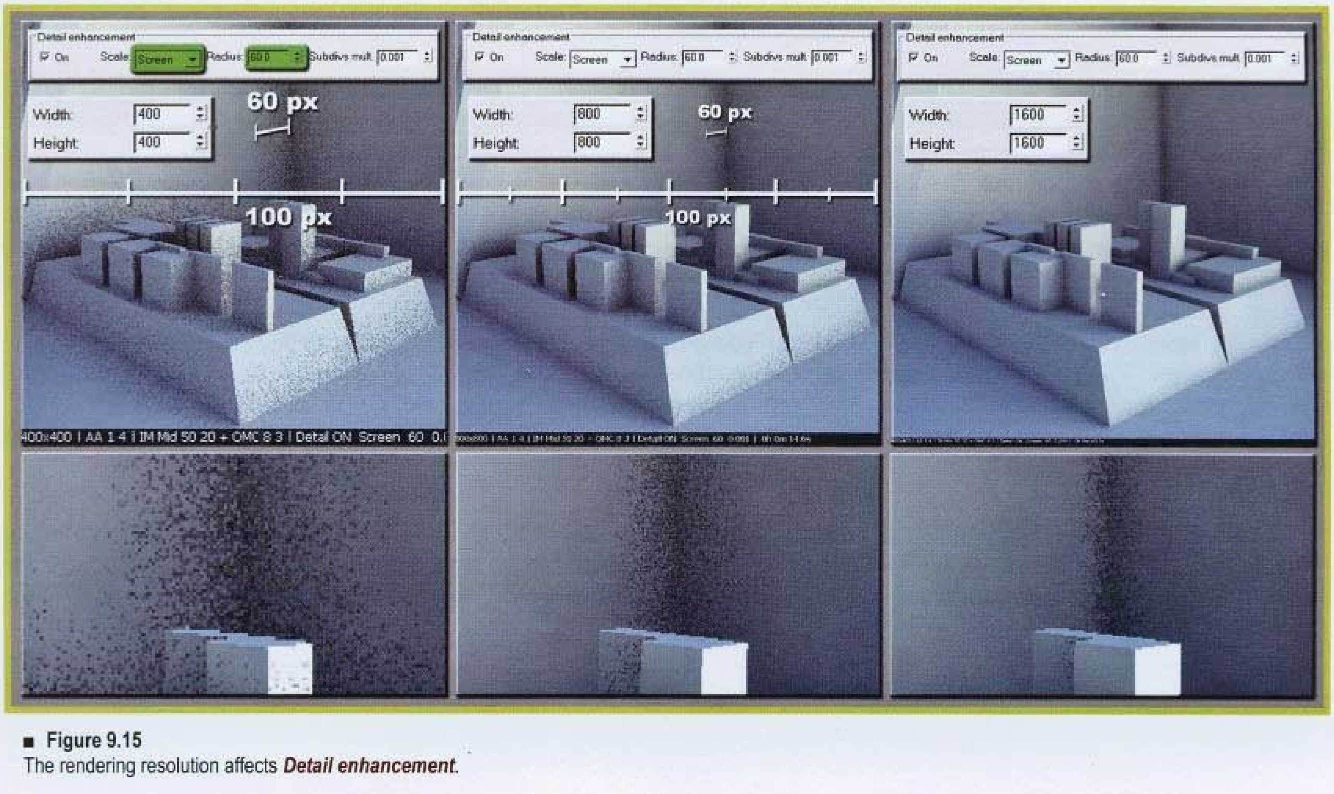


■ **Figure 9.14**
Reduction of samples near the critical geometric zones.

Times have grown and the quality, in truth, has not increased that much. Although this is true, remember you are looking a static image. When accomplishing an animation by using only the ***IM***, flickering of the GI can occur. Furthermore the scene is very simple.

Scale - this parameter allows one to decide the unit of measurement to be used to compute the influence radius of ***Detail enhancement***. Just like the ***Light Cache***, two units exist: ***Screen*** and ***World***.

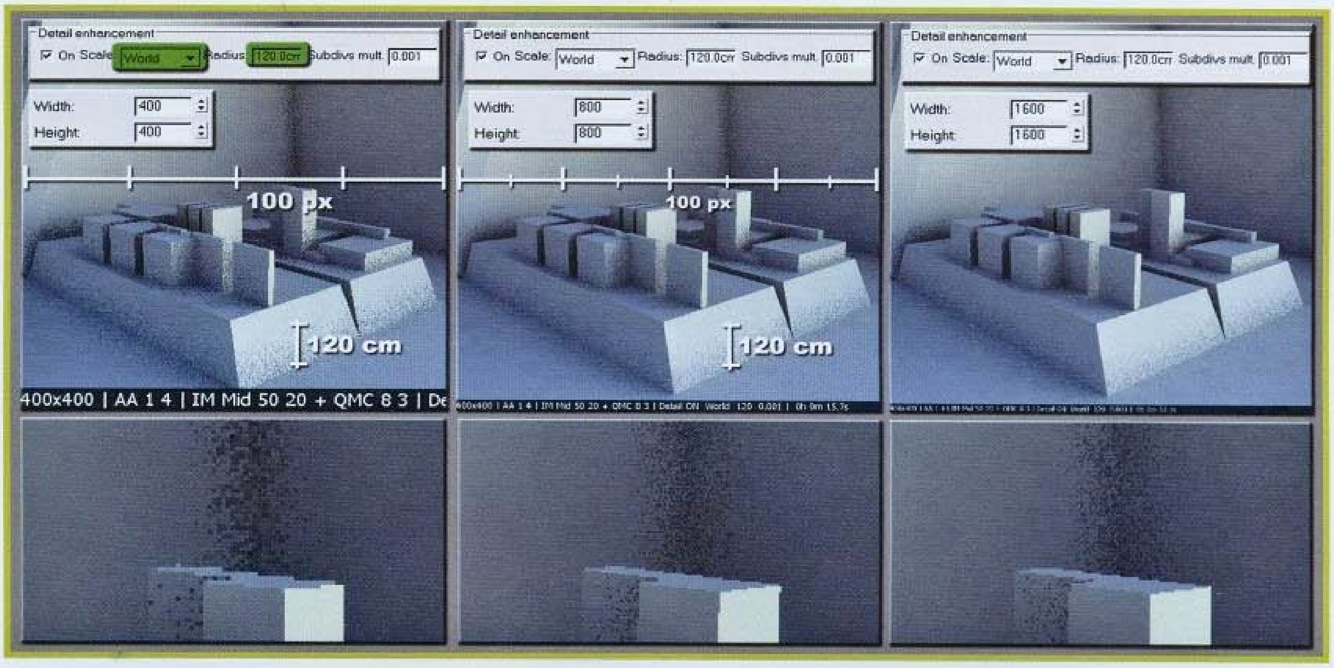
Screen: the mode which defines the influence radius through the pixel unit. Useful in the case of static images. To be avoided when animations are present.



■ **Figure 9.15**
The rendering resolution affects ***Detail enhancement***.

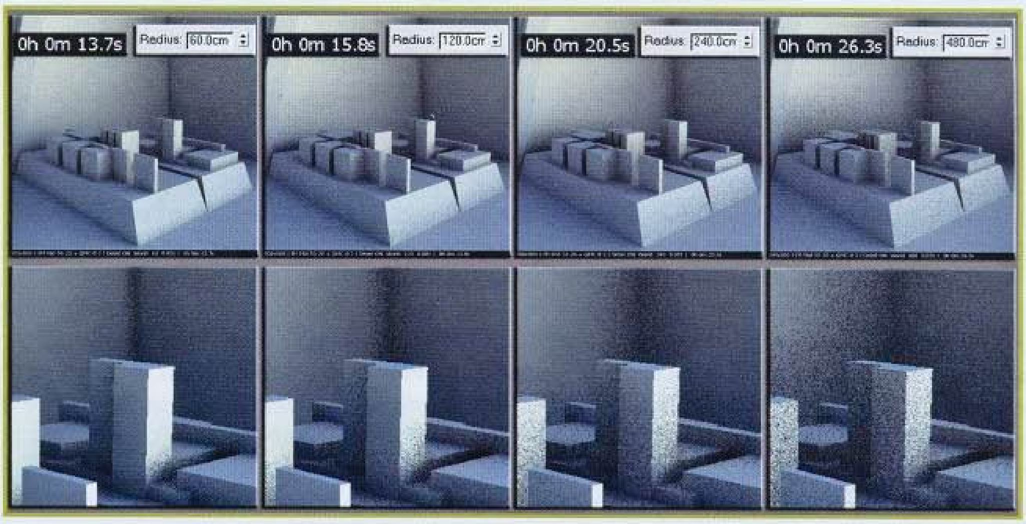
In the previous images default parameters have been used, except for the ***Subdivs multiplier*** option, the parameter controlling the quality of the ***QMC***. Setting it to very low values for educational purposes, the noise of the ***QMC*** is more visible. Furthermore, to make it clearer, ***antialiasing*** has been deactivated. In the first image , rendered at 400x400 pixels of resolution, the influence radius of ***Detail enhancement*** is set to 60 units, that is 60 pixels. As highlighted by the measurement along the vertical corner, the influence radius is roughly 60 pixels. By increasing the resolution twice, the pixels always remain the same, with a consequent reduction of the influence zone of about half, according to the geometric size of the mesh. Doubling the resolution again, up to 1600x1600 pixels, the area halves again, selecting a little vertical band. It has been advised to use this method only for static images. The reason for this is that, since the area is measured in pixels, whether the geometry is near or far away from the camera, the influence zone of ***Detail enhancement*** preserves its size. During an animation it can happen that a wall or a mesh gets closer or further from the observation point. In this case the area of influence would change relative to the geometry, being very small when the geometries are very close. Vice versa, if the same mesh is far away, the extension of the application zone of the ***Detail enhancement*** would be much wider if related to the size of the object. The same situation as the Light Cache, in some way.

World: the mode which defines the influence radius through real units. This parameter is affected by the ***Radius*** parameter like the previous one. ***World*** is a unit to be used in the case of animation. Whether one is near or far away from a geometry or decides to modify the rendering resolution, the influence area remains always the same.



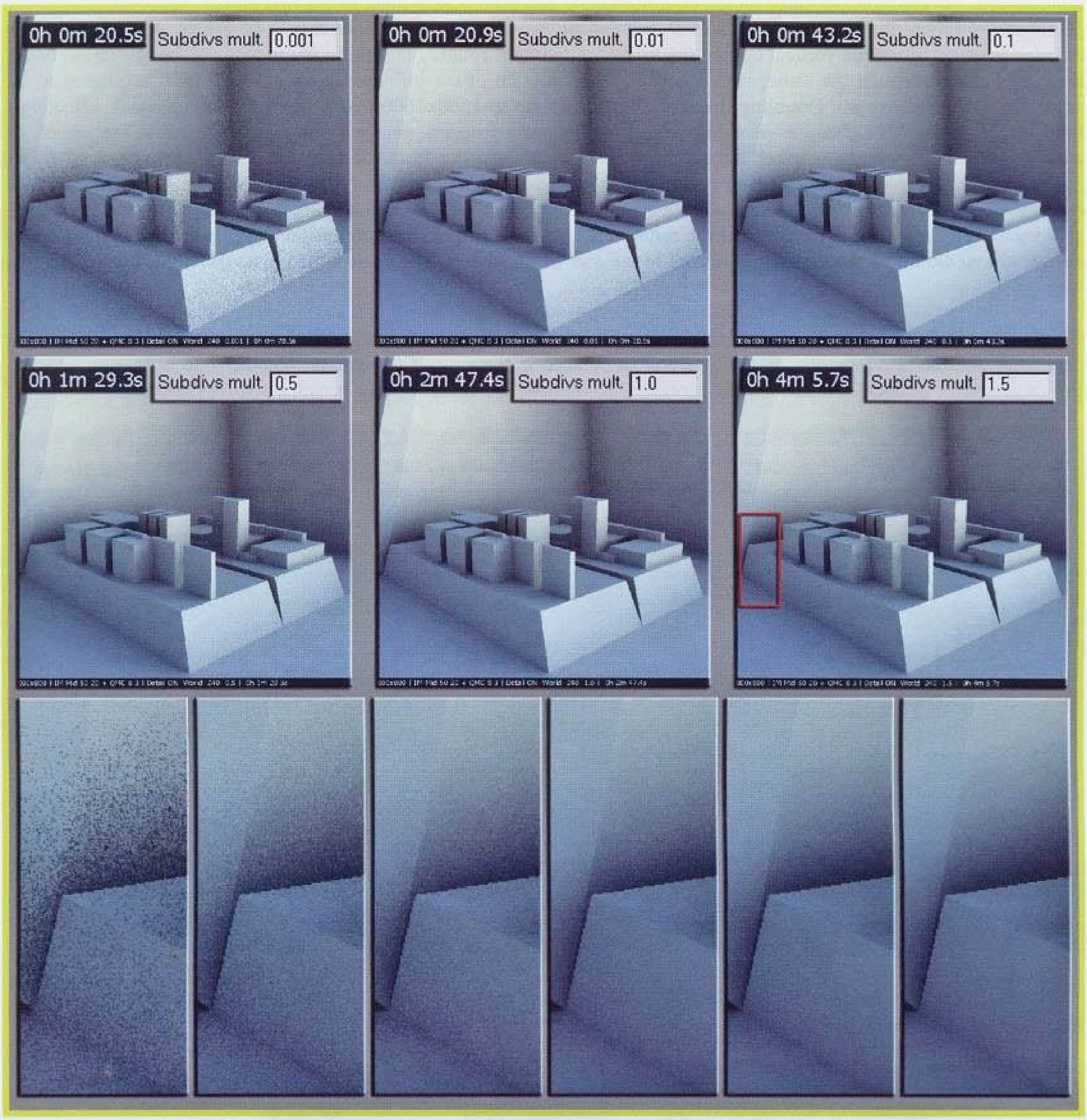
■ **Figure 9.16**
The size of the area of influence is always the same.

Here below, renders with different values of ***Radius***.



■ **Figure 9.17**
Different values of the ***Radius***.

Subdivs mult - with this parameter it is possible to establish the quality, therefore the number of ***Subdivisions*** to be used when computing the ***QMC*** in the ***Detail enhancement***.

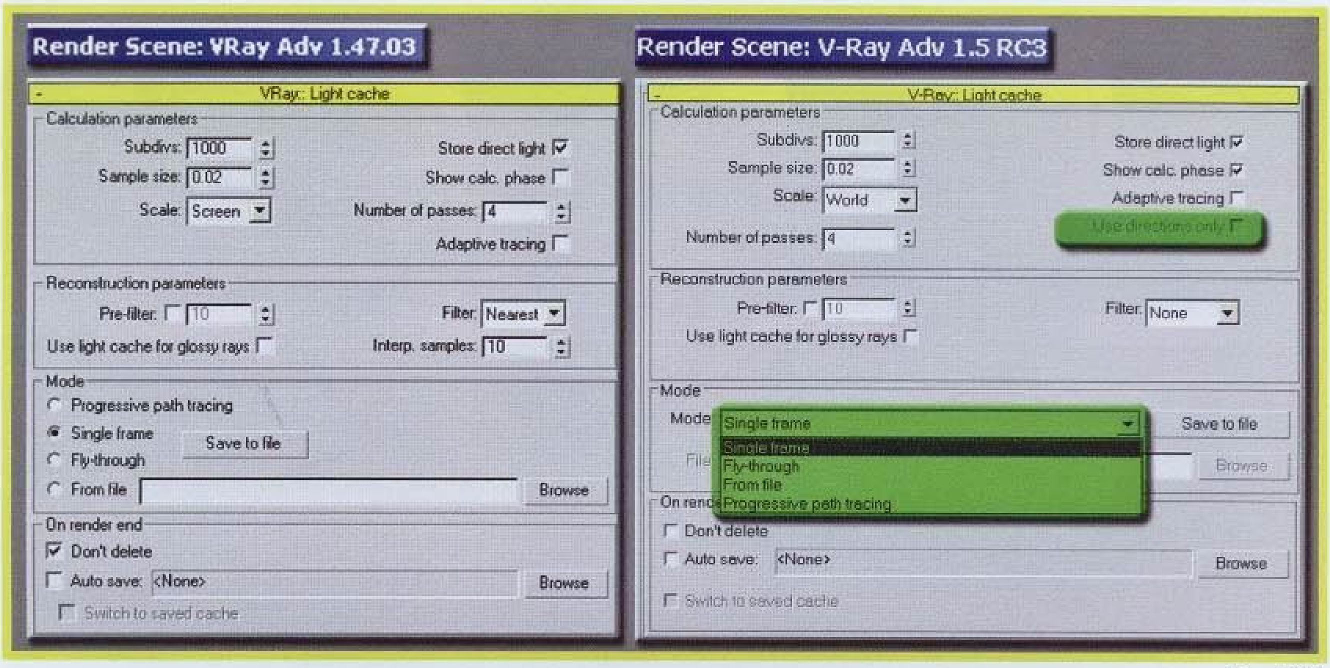


■ **Figure 9.18**
The increase of the quality in the ***Detail enhancement***.
- - - DVD - - -

The term ***Subdivs multiplier*** is made up of two words. ***Subdivision***, classic term indicating the number of subdivisions: the higher this number is, and the higher the quality. Instead, ***multiplier*** indicates that the parameter is a multiplier. In any case remember that one is inside the ***IM***, which has ***HSph Subdivs*** and ***Interp. Samples***.
With the ***Subdivs multiplier*** one defines the quality of the ***QMC*** of the ***Detail enhancement*** as a percentage of the ***HSph. Subdivs*** of the ***IM***. For example, with a value of ***Subdivs multiplier*** =1.0 the number of subdivisions used by ***Detail enhancement*** is the same used by the ***IM***. In this case, since ***HSph. Subdivs*** =50, the subdivisions for the ***QMC*** are also 50: a high value for a brute force system. The default setting of the ***Subdivision multiplier*** is, in fact and not fortuitously, 0.3, that is, one third of the ***HSph Subdivs***. The default number of ***HSph. Subdivs*** is 50, one third is roughly 16 ***subdivs***, which is in the mid-range. Notice that it is possible to use values higher than 1.

Light cache

The *Light Cache* has undergone, like the rollout of the *IM*, a restyling of the interface. The various modes are now inserted in the pull-down menu. Another option has been added, named *Use directions only*.



■ Figure 9.19
The new layout of the *Light Cache*.

Use directions only - the option *Adaptive tracing* allows one to obtain, during the computing of the *LC*, a situation where *VRay* computes more samples where it is necessary, for example in the occluded zones or in areas having lower light. This leads to a greater use of memory, but some effects, such as *GI Caustics,* are affected by a lower level of noise, in the same time. By using the *Use directions only* option only directional information is considered, and not the values of each sample. This makes the calculus unbiased, but potentially less "noisy" relative to the employment of *Sample size* values =0.0. To use these two parameters *Adaptive tracing* and *Use directions only*, in fact, it is necessary to use values of *Sample size* necessarily higher than 1.0.

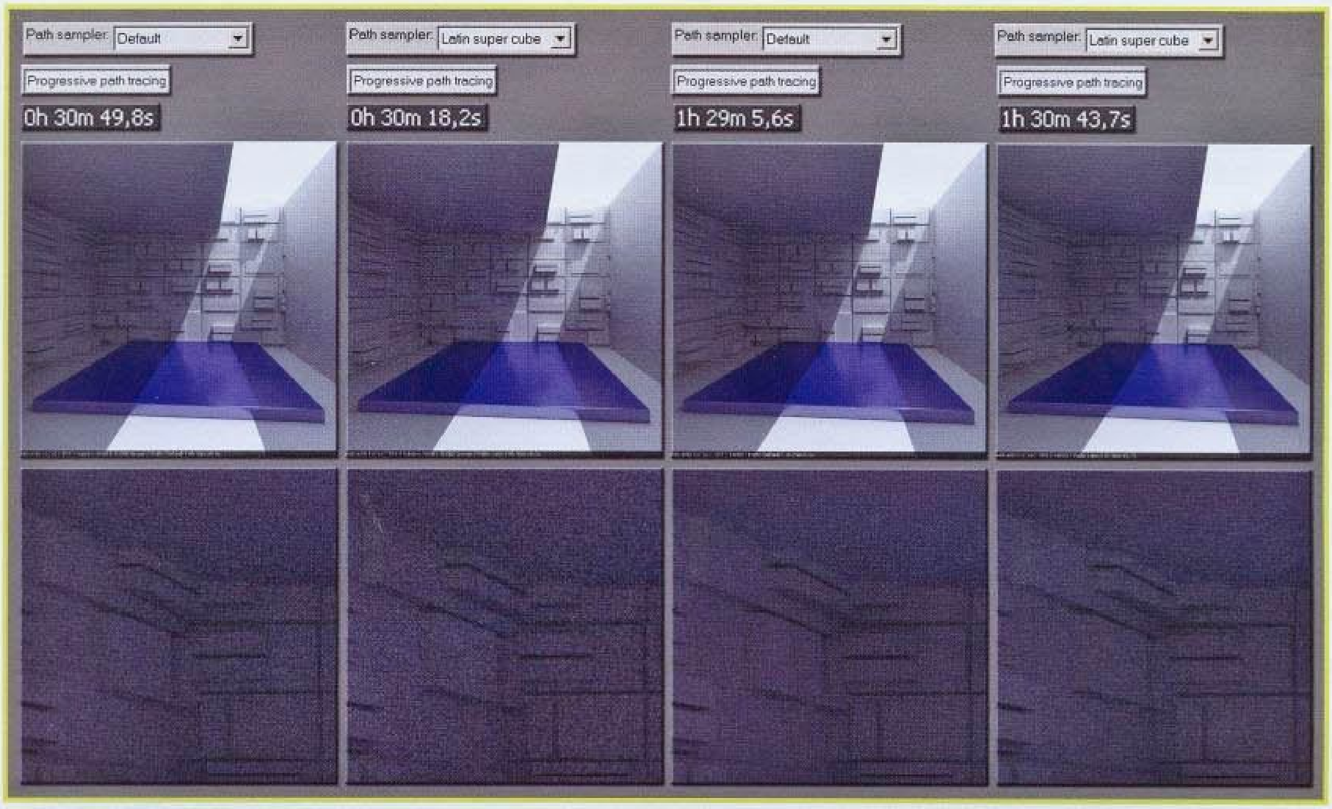
rQMC Sampler

With the new version a little pull-down menu, named *Path Sampler,* has been added in the *VRay: rQMC Sampler* rollout.

■ Figure 9.20
The new option of the *VRay: rQMC Sampler* rollout.

According to the manufacturers, this new computing method for *QMC* functions should solve possible problems of moiré effects during the use of *PPT*.

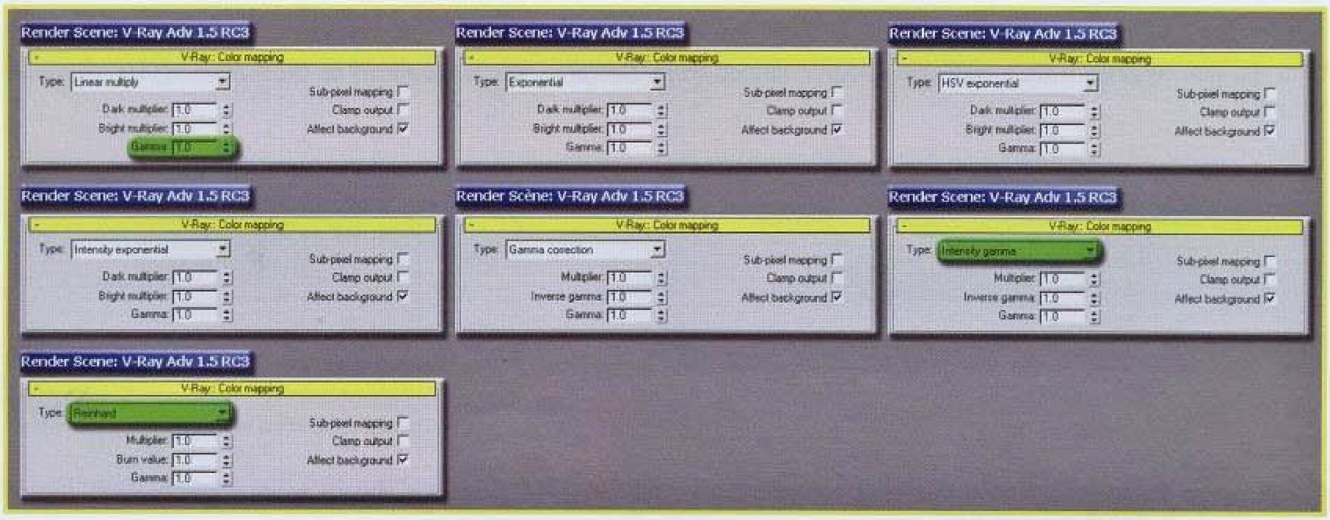
Almost the same rendering times have been obtained in tests carried out in a normal scene. A slight increase of the noise has occurred too.



■ **Figure 9.21**
There are minimal differences by using the ***Latin super cube*** option or not.
--- DVD ---

Color Mapping

In this case as well, the rollout dedicated to ***Color mapping*** has been changed, with corrections to the definitions of the options according to the change of the typology of ***Color mapping***. The ***Gamma*** option has been added too. This parameter is always present, for any mode one chooses. Finally, two modes have been added: ***Intensity Gamma*** and ***Reinhard***. Read the chapter concerning the ***Color mapping*** for an in-depth explanation.



■ **Figure 9.22**
The seven modes of ***Color mapping*** available in VRay 1.5RC3.

VRay 1.5 FINAL

INTRODUCTION

The last version of *VRay* available before the issue of the handbook was released in September 2007. After years of waiting, finally ***v1.5 FINAL*** came out. The main changes carried out in this version are listed, with the analysis, for the most important ones, of their use in real cases.

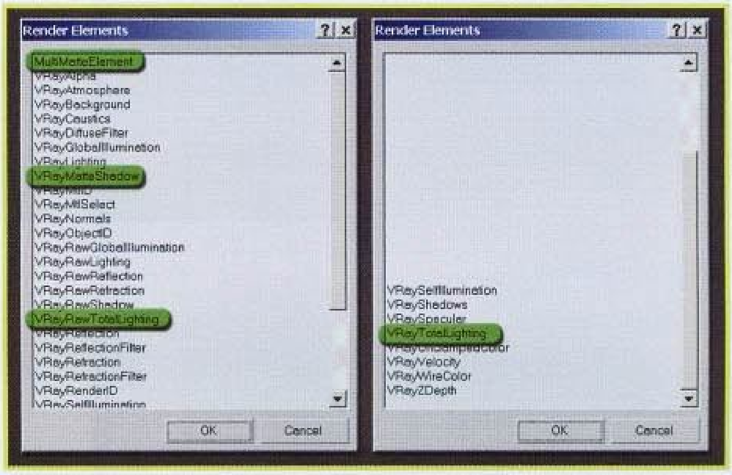
NEW FEATURES	VARIATIONS AND ADDITIONS
Render Elements	New ***VRayMatteShadow***, ***VRayTotalLighting***, ***VRayRawTotalLighting***.
Script	Addition of the export commands in the ***.vrscene*** format in the Quad menu.
Irradiance Map	New ***Animation (prepass)*** and ***Animation (rendering)*** modes.
RENDER ENGINE	
Velocity	Increase of computing speed in particular scenes.
Sampling	Use of the new ***Schlick*** system instead of the ***rQMC***.
INSTALLATION	
Driver	Update of drivers for both the 32 and 64 bit hardware USB key.
Install	Three kinds of installation: ***full workstation***, ***render slave*** and ***license server***.
Uninstall	
LIGHTS AND CAMERAS	
Photometric models	New model for VRaySun, VRaySky, VRayPhysicalCamera and VRayLights.
VRayLights	New parameters: ***Legacy models*** and ***Use 3ds Max photometric scale***.
VRayLights	Addition of the ***Cast shadows*** option.
VRayLights	Addition of the ***Simple portal*** parameter.
VRayLights	Addition of the ***Affect reflections*** parameter.
VRayLights	***Cutoff*** parameter added.
VRayLights	***VRayLights*** produce an accurate ***Motion blur*** if animated.
VRaySun	***VRaySun*** is now integrated in the ***Daylight*** parameter of 3ds Max.
VRaySky	***VRaySky*** uses the first ***VRayLight*** active in the scene.
VRayPhysicalCamera	Addition of the controlling ***vignetting*** parameter.
VRayPhysicalCamera	Addition of the ***horizon line*** parameter.
VRayPhysicalCamera	Addition of the ***clipping plane*** parameter.
VRayPhysicalCamera	Addition of the ***Environment near*** and ***Environment far*** parameters.
DISTRIBUITED RENDER	
DR	Support for the ***Render-to-texture*** system.
DR	Deactivation of servers lacking textures or maps.

GEOMETRIES	
VRayProxy	Faster visualization in the viewport.
VRayProxy	Correct computation of ***Motion blur***.
VRayProxy	Correct computation of ***VRayVelocity***.
Default geometry	The parameter ***Auto*** allowing automatic memory management.
TEXTURES & SHADERS	
Override	Addition of a parameter for the management of the materials in ***VRay: Global Switches***.
Matte	The ***Matte*** objects with the ***Shadows*** option active generate ***VRayRawLighting***, ***VRayRawShadow*** and ***VrayMatteShadow*** channels correctly.
VRayHDRI	Addition of the ***Overall mult*** parameter.
VRayMtl	The ***Roughness*** parameter has been added to the ***VRayMtl***.
VRayMtl	Addition of the computing model ***Hibrid*** to the ***VRayMtl*** for the computing of the ***SSS***.
VRayDirt	Addition of the ***Work with transparency*** parameter.
VRayColor	Addition of a new parameter for changing the ***Gamma*** for a correct visualization inside the Material Editor.
VRayFastSSS	Use of interpolation derived from the IM for a better and quicker result.
VRayEdgesTex	Virtual generation of smooth corners on geometries with "sharp corners".
GI COMPUTATION	
IM, LC e PM	The name of the self-saving file of the ***IM***, ***LC*** and ***PM*** is to be saved at each frame.
IM, LC e PM	The map is to be saved just after it has been computed, instead of at the end of the frame.
LC	The default value of the ***Passes*** for the ***LC*** is 8.
RENDER OUTPUT	
VRay: Color mapping	Addition of the ***Clamp level*** parameter.
VRay: Color mapping	Addition of the parameter ***Don't affect colors***.
OTHERS	
VRayDisplacementMod	Support of ***HDRI*** textures for the ***2D mapping mode (landscape)***.
VRaySphereFade	***Affect alpha*** parameter added.
VRaySphereFade	The ***VRaySphereFade*** parameter works correctly with transparent objects as well.
VRayFur	The ***VRayFur*** generates a correct ***Motion blur*** with more than 2 ***Geometry samples***.
Renderer	The Renderer panel has been divided into: ***VRay***, ***Indirect Illumination*** and ***Settings***.

PARAMETERS

New Render Elements

The new version of *VRay* introduces 3 new Render Elements. These are the ***VRayMatteShadow***, ***VRayTotalLighting*** and ***VRayRawTotalLighting***. A user has also produced a new channel, called ***MultiMatteElement***, that *VRay* programmers have inserted as an integral part of the software.

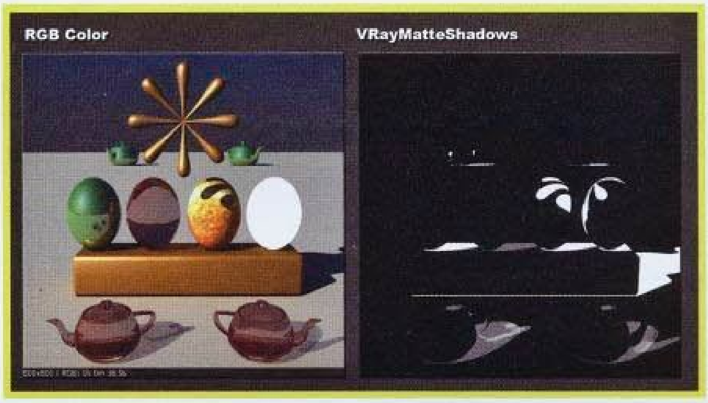


■ **Figure 9.23**
The four new channels.

VRayMatteShadows - this Render Element generates an image similar to an alpha channel. But only shadows are represented with it.

■ **Figure 9.24**
The ***VRayMatteShadow*** channel. Notice the three parameters, familiar by now.

In this case the gray zones identify transparent shadows, while in the white areas the shadows are full.

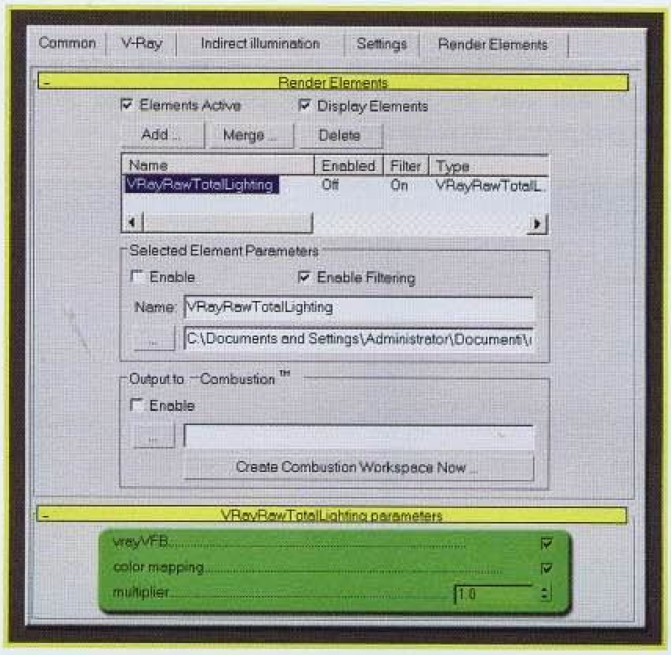


■ **Figure 9.25**
The ***VRayMatteShadow*** layer.

This channel supports:

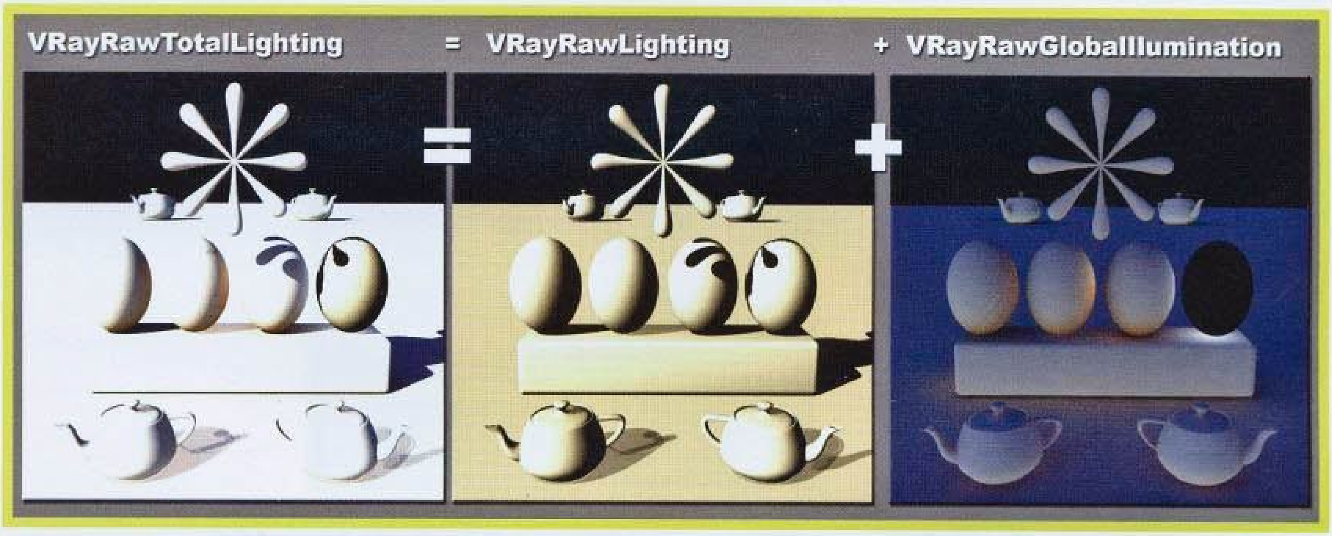
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayRawTotalLighting - As the name suggests, this layer represents the total amount of light present in the scene, without considering textures, shaders, reflections or refractions.



■ **Figure 9.26**
The *VRayRawTotalLighting* layer.

The ***VRayRawTotalLighting*** is the result of the sum between ***VRayRawLighting*** and ***VRayRawGlobalIllumination***.



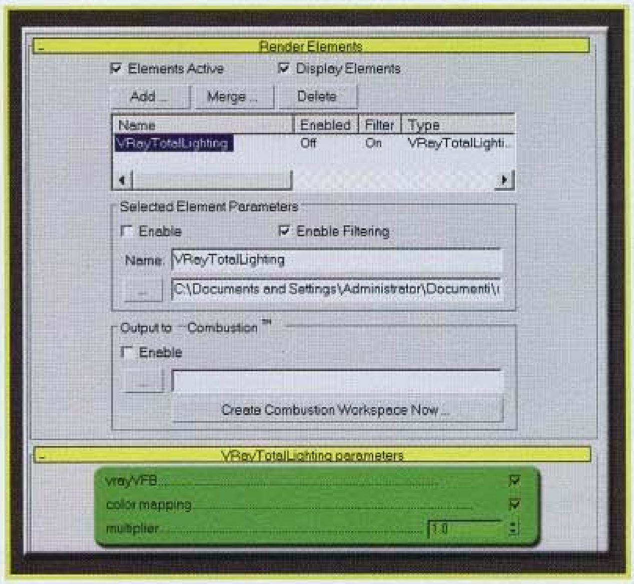
■ **Figure 9.27**
Sum of the two layers.

The ***VRayRawLighting*** channel represents only the direct illumination, as one can see from the completely black shadow generated by the base, whereas the ***VRayRawGlobalIllumination*** layer represents ***Global Illumination*** alone. The union between direct and indirect light allows one to observe the total light.

This channel supports:

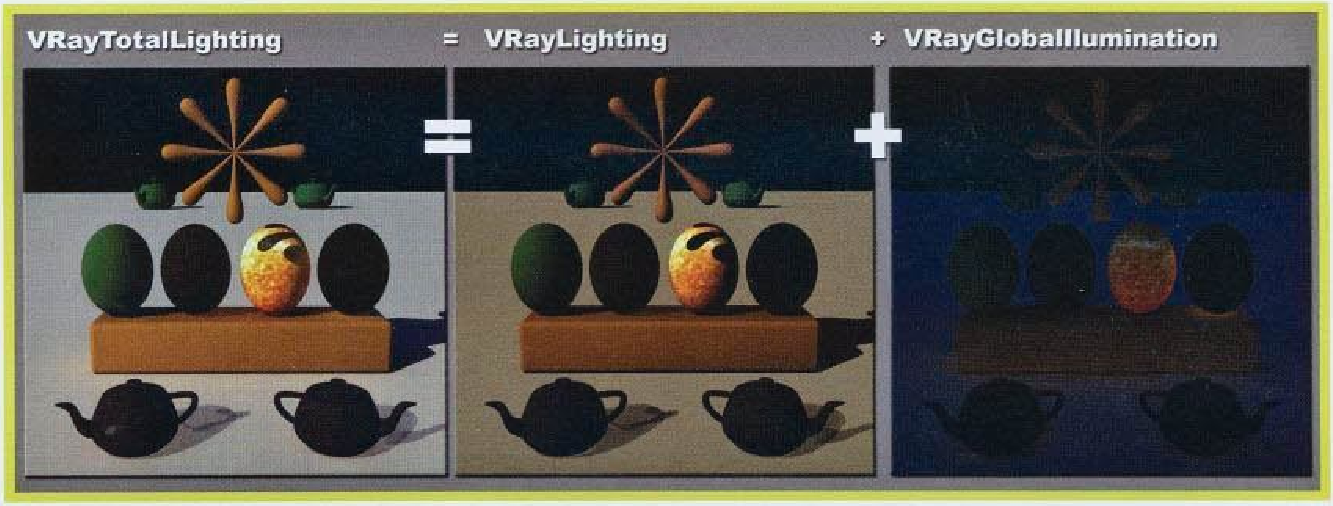
VRayMtl	3ds Max Mtl	Transparency
yes	no	yes

VRayTotalLighting - this layer represents the total amount of light in the scene, considering textures, shaders, reflections and refractions.



■ Figure 9.28
The ***VRayTotalLighting*** layer.

The ***VRayLighting*** channel represents the direct illumination with colors and textures applied, while the ***VRayGlobalIllumination*** represents just the Global illumination. The union of direct and indirect light allows one to observe the total light.

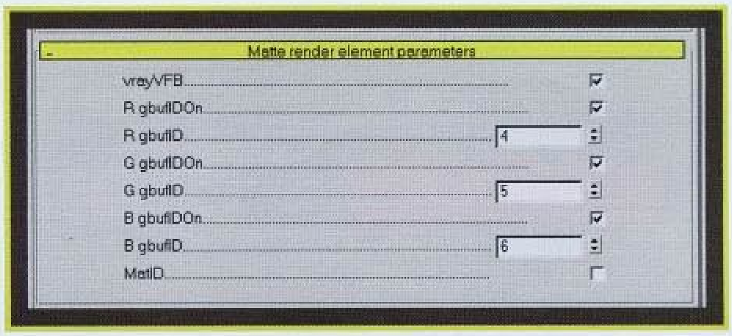


■ Figure 9.29
The sum of the channels allowing to obtain the ***VRayTotalLighting***.

This channel supports:

VRayMtl	3ds Max Mtl	Transparency
yes	yes	yes

MultiMatteElements - this layer allows one to generate the colored ***alpha matte***, according to the ID of the object or the material.



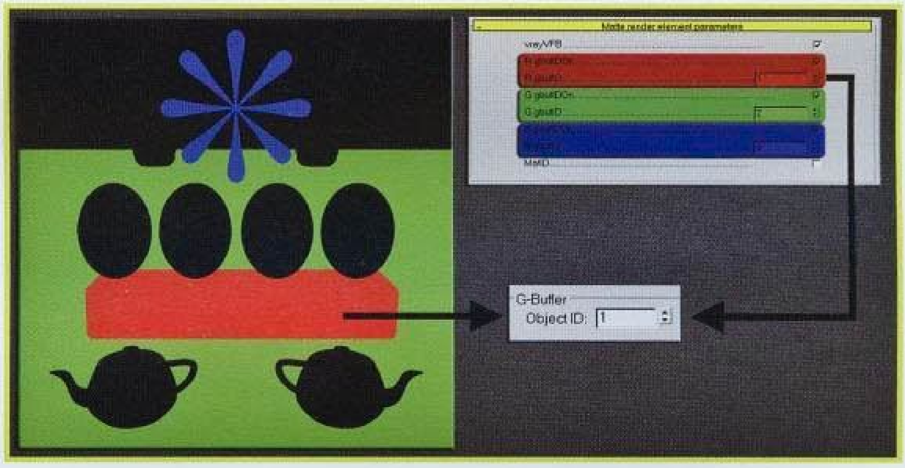
■ Figure 9.30
The ***MultiMatteElements*** layer.

First of all it is worth specifying what is meant respectively by Object ID and Material ID.

■ **Figure 9.31**
The ID Object and ID Material settings for the basement.

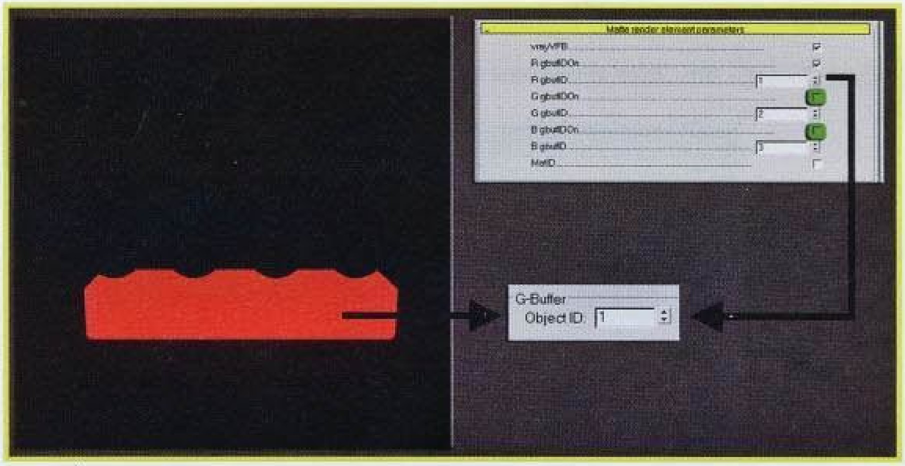
The Object ID is located in the properties of the object, under the G-Buffer. Here it is possible to assign an integer to the geometry, in this case 1. The number is detected by the ***MultiMatteElements*** channel. Instead, the Material ID is present in the Material Editor. All the objects with this material are identified with the same ID. In this example the number associated to the Dark Red material is 2.
The ***MultiMatteElements*** options are very different this time. Except for the ***vrayVFB***, the 7 remaining parameters are completely new. Observe the first example.

■ **Figure 9.32**
The ***MultiMatteElements*** channel.



The layer is represented in four colors: red, green, blue and black. An ID is associated to each color. In the exercise, each object having Object ID 1 is going to have a red color, Object ID 2 corresponds to green and Object ID 3 to blue. If the remaining objects do not have one of these IDs, they are represented in black. It is possible to modify the IDs to be associated to each color with the corresponding R, G, B ***gbufID*** parameter. By deactivating the ***gbufIDON*** parameters just the active channel is to be rendered.

■ **Figure 9.33**
Deactivation of two options for the control of visibility. In this case only the ***RED*** channel is active, with the ***Object ID*** parameter set to 1.



By activating the MatID option, the value inside the Material Editor is considered with integer value instead of the Object ID. An interesting thing is that ***antialiasing*** is active in this case.

This channel supports:

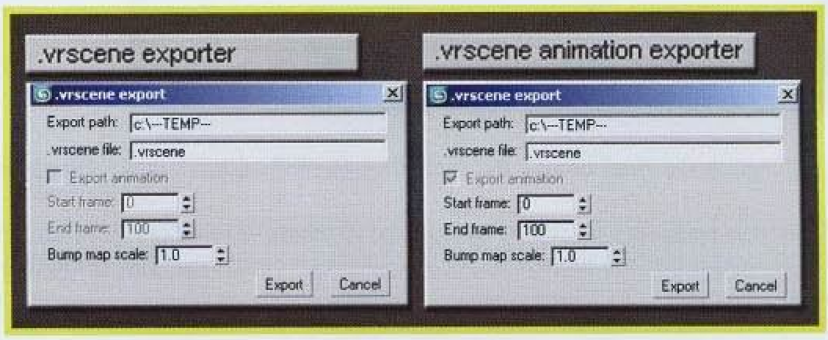
VRayMtl	3ds Max Mtl	Transparency
yes	yes	no

Quad menu

By right-clicking in the viewport the Quad menu appears. With the new version of *VRay* two new scripts have been installed, allowing one to export the scene form the .max format to the ***.vrscene*** format, which is supported by the standalone of *VRay*. Up to this date, this version is still being developed.

■ **Figure 9.34**
The Quad menu and the two scripts installed by ***VRay v1.5 FINAL***.

It is possible to export both a single file and an animation.



■ **Figure 9.35**
The exportation windows.

Export path - it specifies the saving path. The directory has to be created before exporting. Otherwise an error occurs in the script.

.vrscene file - it specifies the name of the save file.

■ **Figure 9.36**
The two exported files.

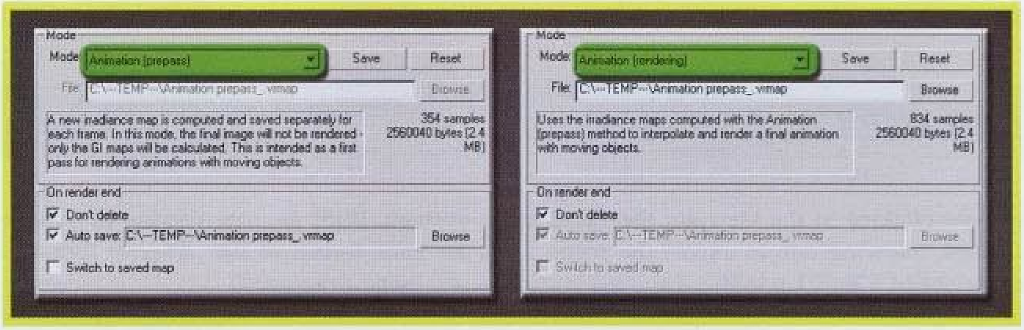
Start and End frame - in the case of exportation of an animation, it specifies the number of frames to be exported.

Bump map scale - it specifies the intensity of the exported Bump channel. Since the standalone version of *VRay* is not available, it is impossible to carry out tests or comparisons to verify the exact meaning of this parameter.

Export - it starts the exportation process.

Cancel - it stops the exportation process.

VRay introduces two new computing modes with the ***VRay 1.5FINAL*** version: ***Animation (prepass)*** and ***Animation (Rendering)***. These methods are used to reduce flickering problems in dynamic animations.

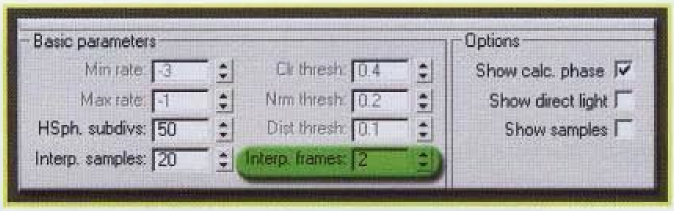


■ **Figure 9.37**
The two new modes for computing the ***IM***.

A dynamic animation is, by definition, a scene where objects, colors, textures, etc... change at each frame. For this reason the GI must be computed in each instant. With the ***Animation (prepass)*** *VRay* computes the ***IM*** for each single frame and saves it in an corresponding file. This way, if the sequence is formed by 100 frames, *VRay* generates the same number of irradiance maps.

■ **Figure 9.38**
The ***IM*** files generated for each single frame.

VRay does not automatically generate the final image, but computes the ***IM*** only, as if the ***Don't render final image*** were activated.
Once the calculation is finished, it is necessary to switch over to the ***Animation (rendering)*** mode. *VRay* demands the irradiance maps it requires. In this case, the 100 maps which have just been computed. The strength of this method lies in the ***Interp. frames*** parameter.



■ **Figure 9.39**
The ***Interp. frames*** parameter for mixing several Irradiance Maps.

Thanks to this option it is possible to establish how many frames *VRay* has to use for computing the GI. In practice, for rendering a specific frame, *VRay* does not use just the GI of that specific frame, but it mixes the GI of other frames. With the blending of multiple irradiance maps flickering problems are further reduced.

Animation (prepass) - in this mode the ***IM*** is created for each frame and saved in a separate file. This way *VRay* renders a map for each frame. Therefore, it is not possible to compute every Nth frame. *VRay* automatically disables the render for the final image, computing the ***IM*** only.

Animation (rendering) - in this mode *VRay* renders the final animation by using the ***IM***s created with the ***Animation (prepass)*** method. The ***IM***s obtained by adjacent frames are loaded and mixed. The number of frames to consider is managed by the ***Interp. frames*** parameter.

Interp. frames - this parameter defines the number of frames to be used when the ***Animation (rendering)*** mode is used. The default value 2 guarantees the use of 5 frames: the one being rendered, the two previous ones and the two following ones. High values slow down the calculation of the render, providing more uniform animations, but lacking in details.

Sampling

Like one may have noticed, inside ***VRay 1.5FINAL*** the word ***rQMC*** (random Quasi Monte Carlo) has been substituted by the ***DMC*** (Deterministic Monte Carlo). The ***DMC*** is a version of the famous Monte Carlo sampler. The main difference is that, while in the pure Monte Carlo, when two renders of the same scene are computed, the noise within it is always slightly different, in the ***DMC*** this does not happen. Furthermore, the ***DMC*** is very quick and produces less noise compared to the previous versions when scenes with a lot of blurry effects, such as ***Area Shadows***, ***DOF***, ***Motion Blur***, ***GI*** etc... are involved. By default the ***Schlick*** sampler is used, introduced in 1991 by Christophe Schlick. In any case, both the old system, now called ***Legacy***, and the ***Latin super cube*** system, mainly introduced to reduce the possible reproducible moiré effect problems of the ***Legacy*** method, remain available.

■ **Figure 9.40**
The samplers available for the ***DMC***.

In the following examples 3 renders are compared. They have been fulfilled by using:

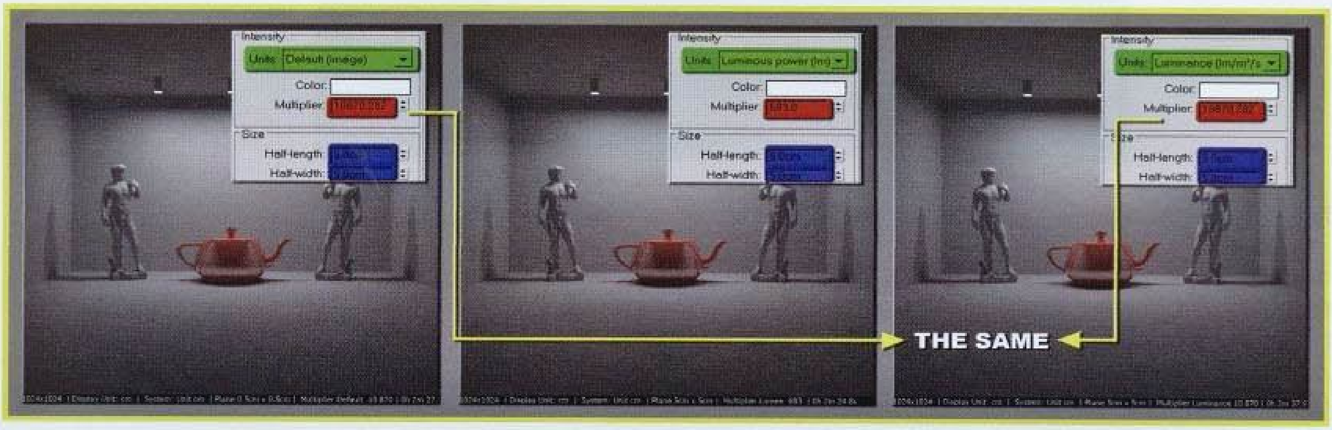
Antialiasing: ....***Adaptive DMC Min subdivs*** = 1 ***Min subdivs*** = 16.
GI: ....***Brute force*** + ***Brute force Subdivs*** = 8 ***Secondary bounces*** = 3
DOF: ....***Subdivs*** = 6
Area Shaows:... ***Subdivs*** = 3
Glossy reflect: ...***Subdivs*** = 8

■ **Figure 9.41**
Difference in noise accomplished with the three types of samplers.

The scene has many glossy effects. Particularly, the employment of the ***Schlick*** sampler has allowed one to obtain a glossy with less noise in the same time spent by the other two methods. The highest rendering time has been reached with the ***Latin super cube*** system, producing more noise.

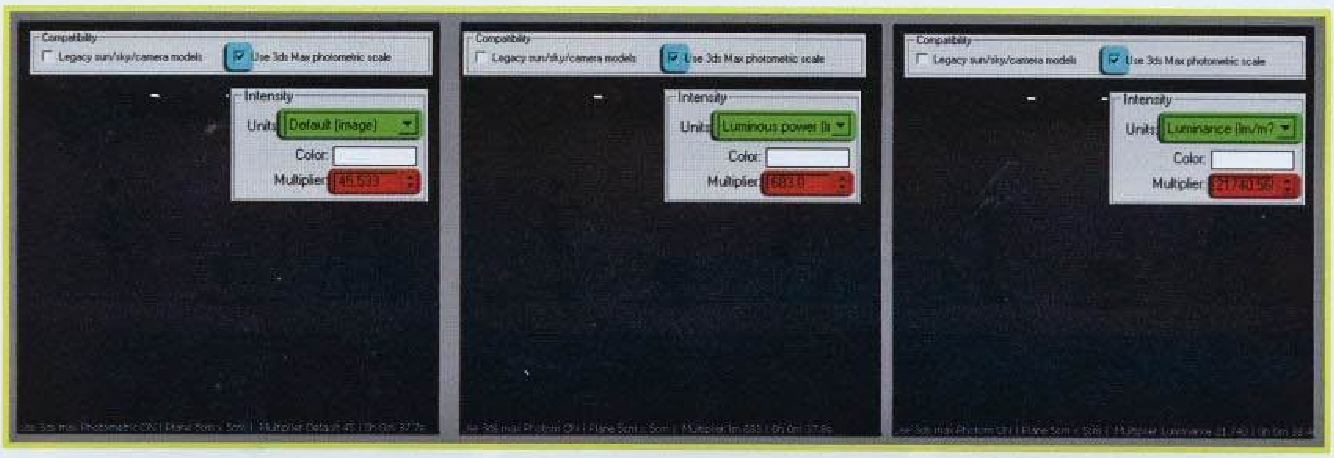
Legacy models and Use 3ds Max photometric scale

In the chapter concerning the units of measurement of the ***VRayLight*** we have said that some modifications to the conversion values of the ***Multipliers*** of the lights would have been carried out in the ***v1.5 FINAL*** version of ***VRay***, in order to obtain physically more correct results. In fact, before these corrections, ***VRay*** did not reach an exact correspondence between the units of measurement used in ***VRayLight***, ***VRaySun***, ***VRaySky***, ***VRayPhysicalCamera*** and reality. Therefore some changes have been applied. To better understand this idea, observe the following examples fulfilled with the ***v1.49***:



■ **Figure 9.42**
Results obtained with the ***v1.49*** of VRay.

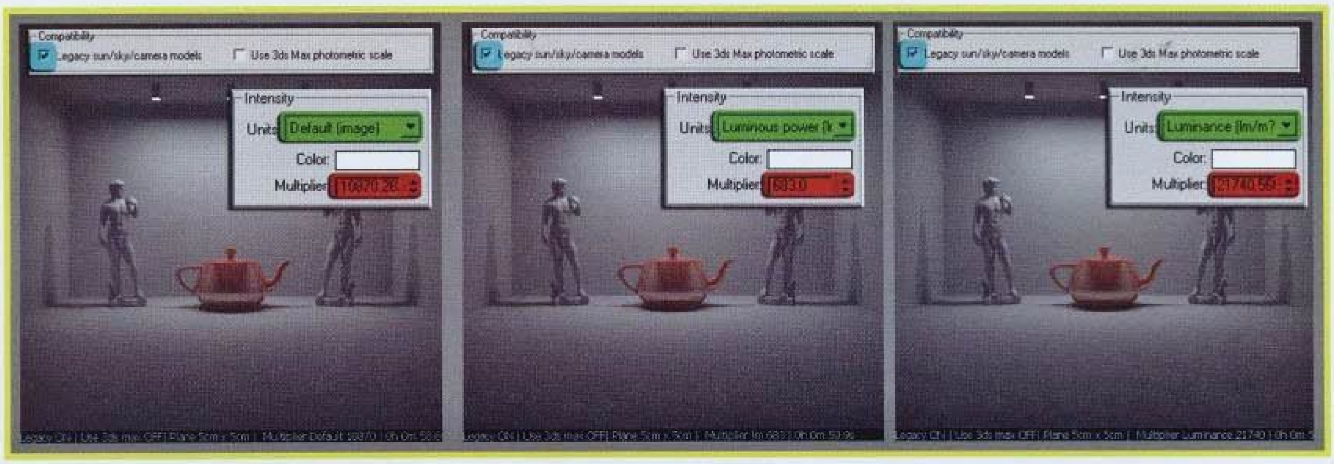
The conversion between 683 ***Lumen***, the ***Default*** unit, and ***Luminance*** would lead to a value of 10.870 lm/m2/sr. Now ***VRay*** has added two options which interact directly with ***VRayLight***, ***VRaySun***, ***VRaySky***, and ***VRayPhysicalCamera***. The default settings, the ones for a physically correct rendering, expect the ***Use 3ds Max photometric scale*** parameter to be active. The next renders, computed with the ***v1.5 FINAL***, are generated by the opening of the previous scene accomplished with the ***v1.49*** version. The only operation is to activate ***Use 3ds Max photometric scale***.



■ **Figure 9.43**
The same values of lumen used in the ***v1.5 FINAL*** of VRay return completely different results.

The scene uses the same lumen values, but the result is completely different. This happens because now ***VRay*** computes the luminous flux of the ***VRayLight*** in a different way. Now the ***Legacy sun/sky/camera models*** is activated, which allows a retro-compatibility with the scenes accomplished with the versions previous to the ***v1.5 FINAL***. This affects the behaviour of the ***VRayPhysicalCamera*** as well. The conversion between lumen and the remaining units changes as well:

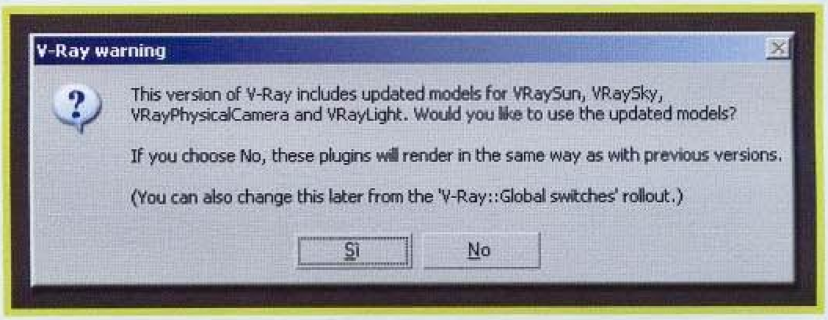
Default Unit x VRayLight Area x 2π = lm Unit
Luminance x VRayLight Area x π = lm Unit
Radiant power / VRayLight Area / π = Radiance



■ **Figure 9.44**
Activation of the ***Legacy sun/sky/camera models*** parameter.

The result is correct. Or, better, it is the same to the one obtained with the versions prior to ***v1.5 FINAL***. This phenomenon recurs in many of the parameters concerning ***VRayLight***, ***VRaySun***, ***VRaySky***, and ***VRayPhysicalCamera***. Summarizing, in order to maintain the compatibility and obtain renders generated with files accomplished with versions preceding the ***v1.5 FINAL***, the options ***Legacy sun/sky/camera models*** must be activated, by deactivating the ***Use 3ds Max photometric scale***. Vice versa, for the new scenes rendered with the ***v1.5 FINAL***, the ***Use 3ds Max photometric scale*** parameter is active by default, whereas ***Legacy sun/sky/camera models*** is switched off.

When files preceding the ***v1.5 FINAL*** are opened, *VRay* makse a notification message appear:



■ **Figure 9.45**
Message appearing when a file generated with the ***v1.49*** version of VRay is opened.

With this message the user is told that the current version of *VRay* installed within 3ds Max, ***v1.5 FINAL***, includes updates such as ***VRaySun***, ***VRaySky***, ***VRayLight***, ***VRayPhysicalCamera***. Therefore the user is asked if he wants to update the ***VRaySun***, ***VRaySky***, ***VRayLight***, ***VRayPhysicalCamera*** present in the file which is being opened with the new settings. If you answer NO, *VRay* uses the ***Legacy*** system, producing a render equal to the one generated, in this case, with the ***v1.49*** of *VRay*. In the opposite case, the ***Use 3ds Max photometric scale*** is used. This choice is not binding and it is possible to modify it moreover by manually activating or deactivating the ***Legacy sun/sky/camera models*** and ***Use 3ds Max photometric scale*** options.

Legacy sun/sky/camera models - in the versions of *VRay* prior to ***v1.5 FINAL***, the computing of ***VRaySun***, ***VRaySky*** and the ***VRayPhysicalCamera*** was not completely correct from the physical point of view. If this parameter is deactivated, which happens by default, *VRay* solves this problem. If it is active, *VRay* modifies its behaviour to be used in scenes preceding ***v1.5 FINAL*** too.

Use 3ds Max photometric scale - when active, by default, this parameter changes the behaviour of ***VRaySun***, ***VRaySky*** and the ***VRayPhysicalCamera*** to conform to the photometric units of 3ds Max. If it is deactivated, *VRay* operates in its internal photometric system, like in the older versions. By activating this parameter one makes sure that the ***VRayLights*** are going to behave in the same way as the 3ds Max ones.

Observe the first scene. It is the previous file, set with ***Legacy sun/sky/camera models*** active. The ***VRayLights*** are set to 683 lm. In the same way, 3 photometric lights have been created, them too set to 683 lm.

■ **Figure 9.46**
Activation of the old photometric management system. The ***VRayLights*** and the photometric lights of 3ds Max have the same multiplier parameters, but the result is not the same at all.

Even though they have same intensity, the ***VRayLights*** and the photometric lights of 3ds Max do not produce the same result. By activating the ***Use 3ds Max photometric scale*** parameter things change radically.

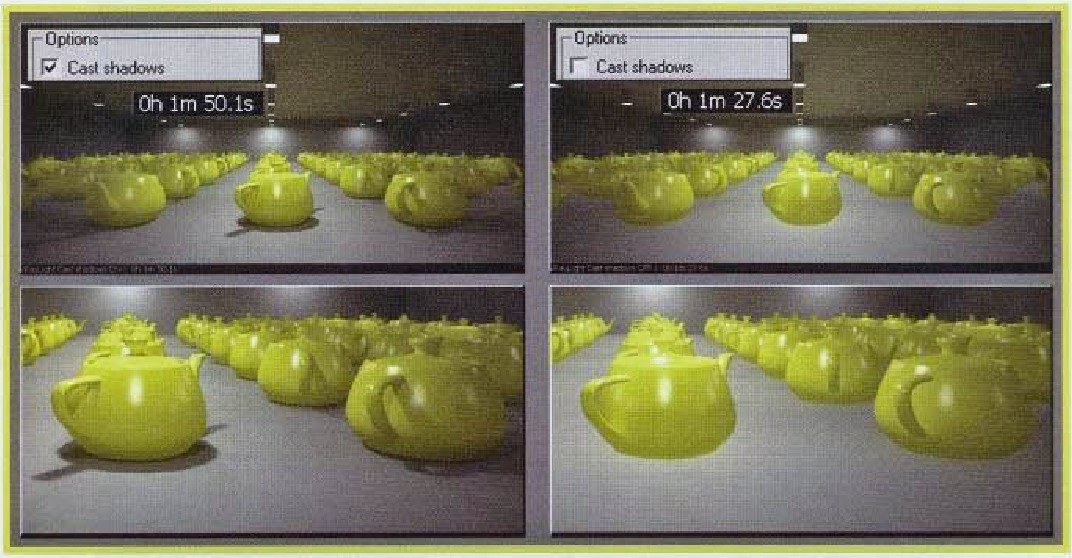
■ **Figure 9.47**
Activation of the new photometric management system. The ***VRayLights*** and the photometric lights of 3ds Max have the same multiplier parameters, and the result is the same.

VRayLight

With the new ***v1.5 FINAL*** some interesting features have been added to the ***VRayLights***. They are the ***Cast Shadows***, ***Simple portal***, ***Affect reflections*** and ***Cutoff***.

■ **Figure 9.48**
The new parameters of the ***VRayLights***.

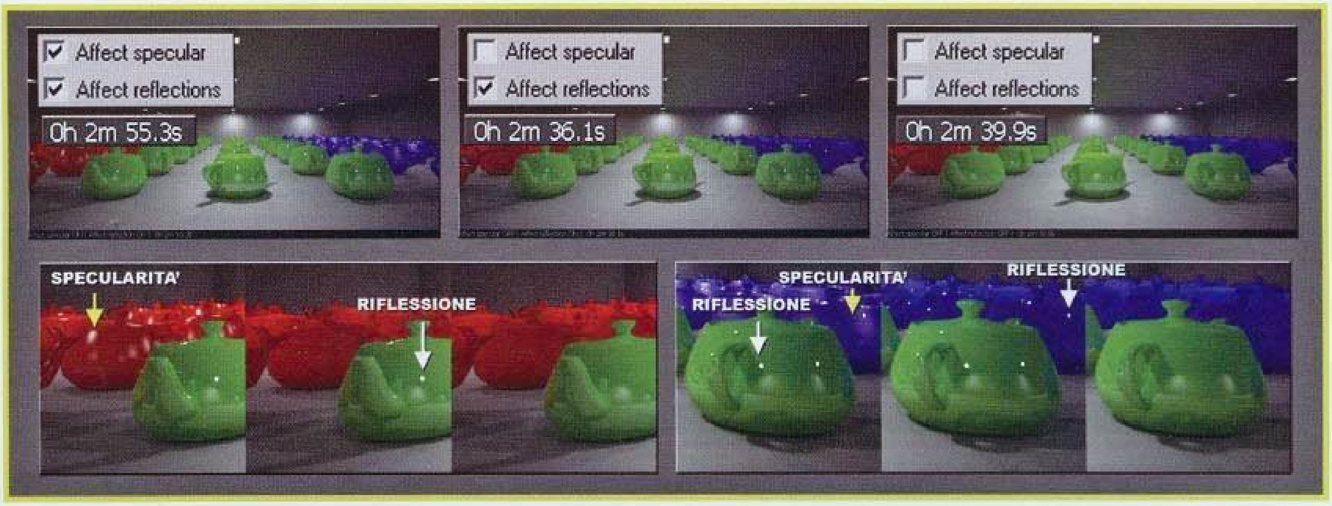
Cast shadows - with this parameter one controls the visibility of the shadows during the rendering process. If it is deactivated, shadows are not be generated.



■ **Figure 9.49**
The deactivation of shadows leads to a high decrease in computing time.

Simple - this parameter is available if the ***Portal*** option is active. The ***VRayLights***, which are usually located on the windows, use the light source present behind them for illuminating the scene, or, in any case, they analyze the space behind them to use the information obtained in order to produce a luminous flux. A certain quantity of rays must be used for studying the elements to be projected. This slows down the rendering process. By activating the ***Simple*** option, ***VRay*** ignores everything behind the ***VRayLights*** and uses exclusively the color or the texture of the Environment, speeding up the render.

Affect reflection - by activating this parameter, ***VRayLights*** are going to be visible in the reflections of objects. Notice how the visibility of the specularities generated by the Highlights and the real reflections are now separately managed.



■ **Figure 9.50**
Different examples of employment of the ***Affect specular*** and ***Affect reflections*** parameters.

In this scene three materials have been used. The first one, the red one, has only HighLights. The green tea pots reflect only, while the blue ones, thanks to the ***VRayBlendMtl***, generate both the HighLights and reflections.
By deactivating the ***Affect specular*** parameter, the point of maximum illumination on the tea pots disappears. By deactivating reflections, the ***VRayLights*** do not reflect anymore.

Cutoff - this parameter specifies the intensity threshold of ***VRayLights*** below which no more light is produced. This parameter can be useful in scenes with lots of lights, where one wants to limit their influence just to a certain distance from the source. High values reduce the influence, speeding up the render.

■ **Figure 9.51**
With a very low ***Cutoff*** value, ***VRayLights*** have a very high influence and the luminous flux takes very far objects into account. By increasing the threshold, the ***VRayLights*** reduce their effects to closer zones. With values near to 1.0 the ***VRayLights*** illuminate very close zones. The rendering times drastically diminish while the ***Cutoff*** increases.

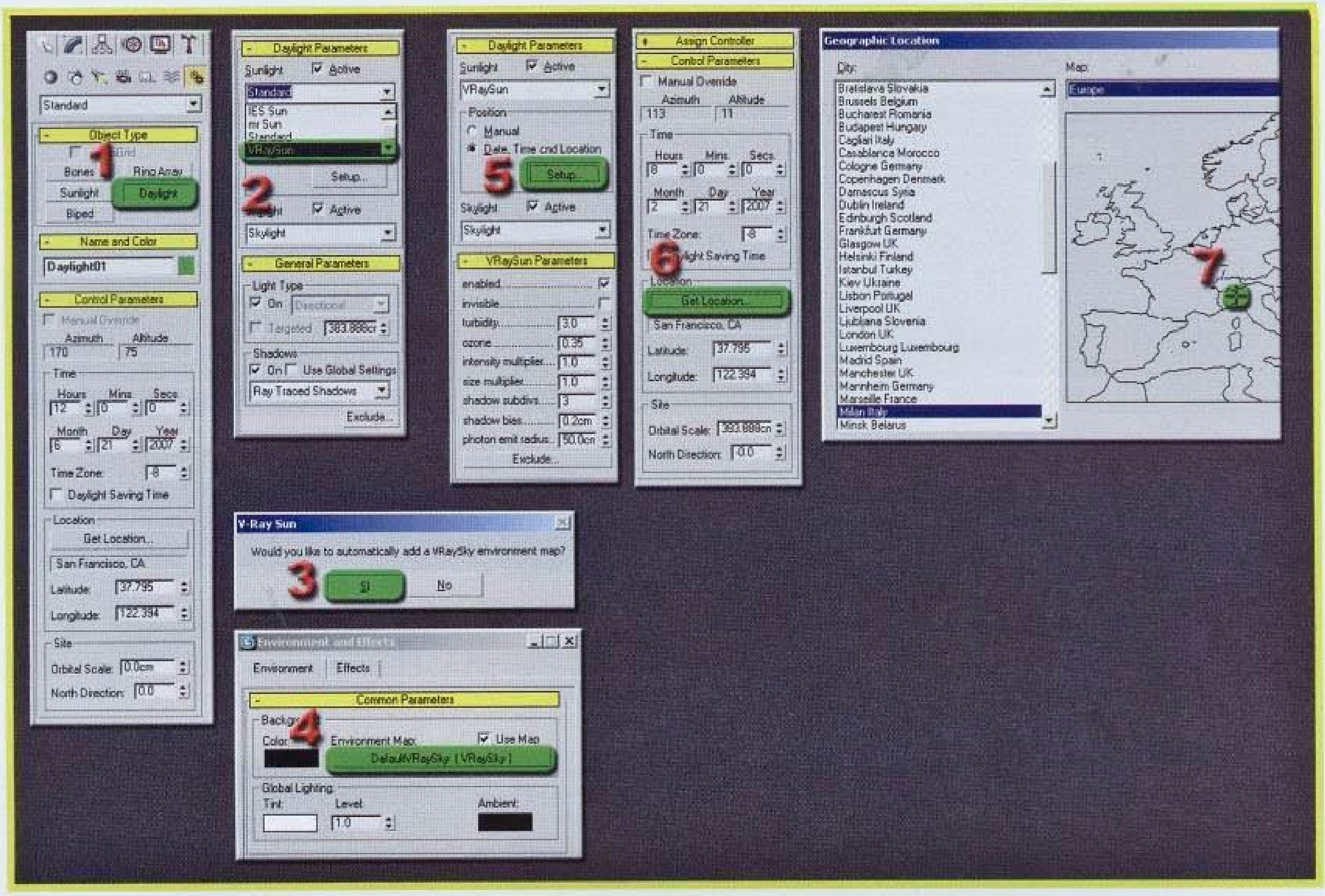
VRAYLIGHT AND MOTION BLUR

Thanks to the improvements of the last version, now the ***VRayLights***, if animated, are able to produce an accurate Motion blur.

■ **Figure 9.52**
The ***VRayLights*** are able to produce a correct Motion blur, as highlighted in the respective alpha channels.

VRAYSUN and DAYLIGHT

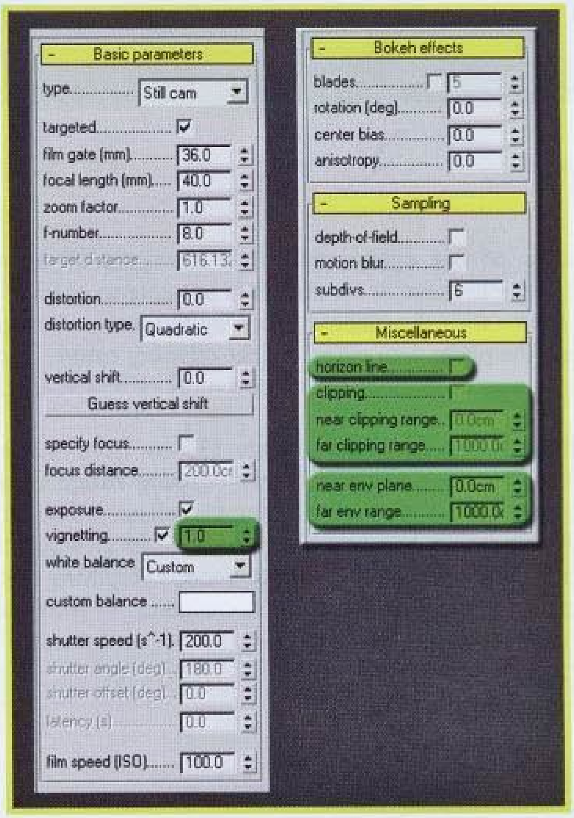
With ***v1.5 FINAL*** of ***VRay*** full compatibility with the Daylight system of 3ds Max has been reached. The Daylight of 3ds Max allows one to generate a physically correct Sun/Sky system with which it is possible to set the geographic coordinates, that is, the precise longitude and latitude, for realistic simulations.
Once the Daylight system (1) has been created, one can choose the ***VRaySun*** (2) from the pull-down menu, to simulate the sun. It is automatically asked if one wants to use the ***VRaySky*** as Environment. The answer is, 99% of the times, yes (3). This way, ***VRaySky*** (4) is added in the Environment of 3ds Max. Now everything is ready. It is enough o select the Daylight System (5), choose the geographic locality, and click on the geographic map. It is necessary to remind that a ***VRayPhysicalCam*** needs to be used for observing the scene has to be created in order to render correctly with the ***VRaySun*** active.



■ Figure 9.53
The steps for the creation of a Daylight system, by using the ***VRaySun*** and ***VRaySky*** of VRay.

VRayPhysicalCamera

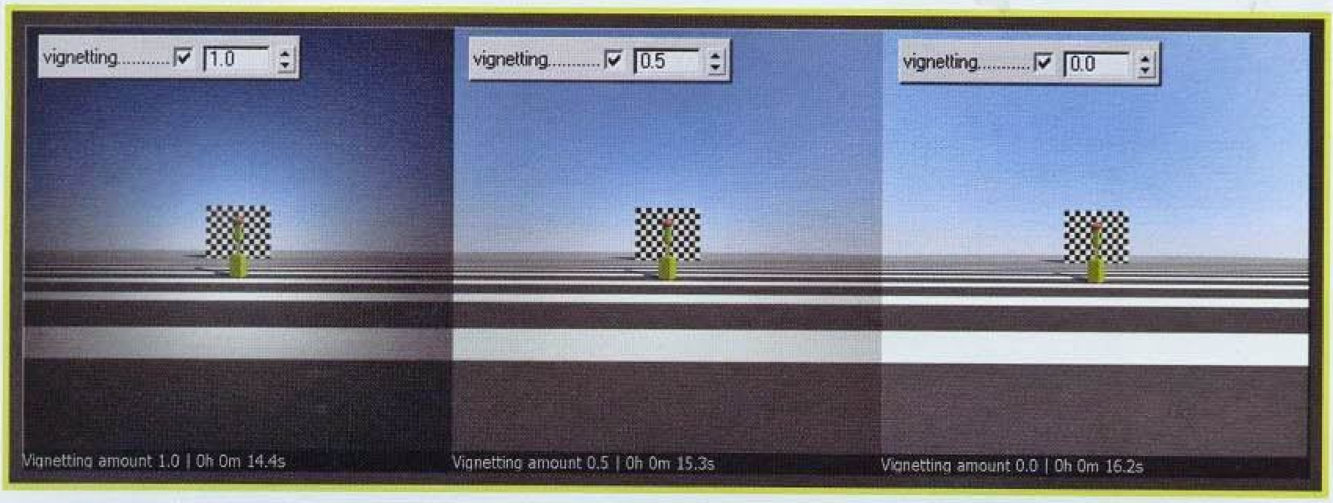
Like ***VRayLights***, some new features have been added to the ***VRayPhysicalCam*** too.



■ **Figure 9.54**
Image 9.54 The added parameters to the ***VRayPhysicalCam*** are:

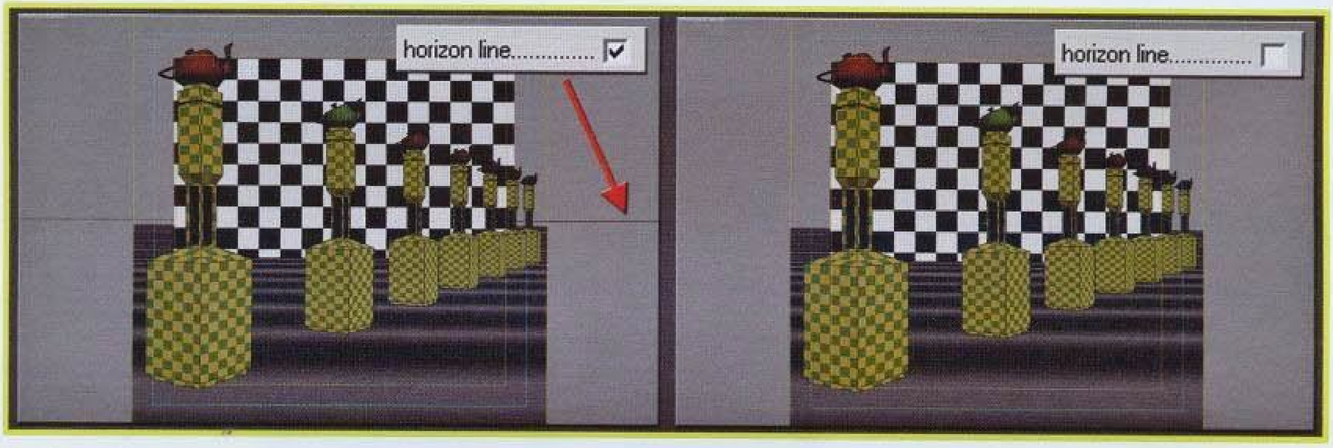
. ***Vignetting amount***
. ***horizon line***
. ***clipping***
. ***environment near and far.***

vignetting amount - it allows one to specify the amount of ***vignetting*** present in the rendering. With ***vignetting amount*** = 0 no effect is present, while with ***vignetting amount*** =1, the effect is the normal intensity.



■ **Figure 9.55**
With the spinner near the ***vignetting*** parameter it is possible to control the effect.

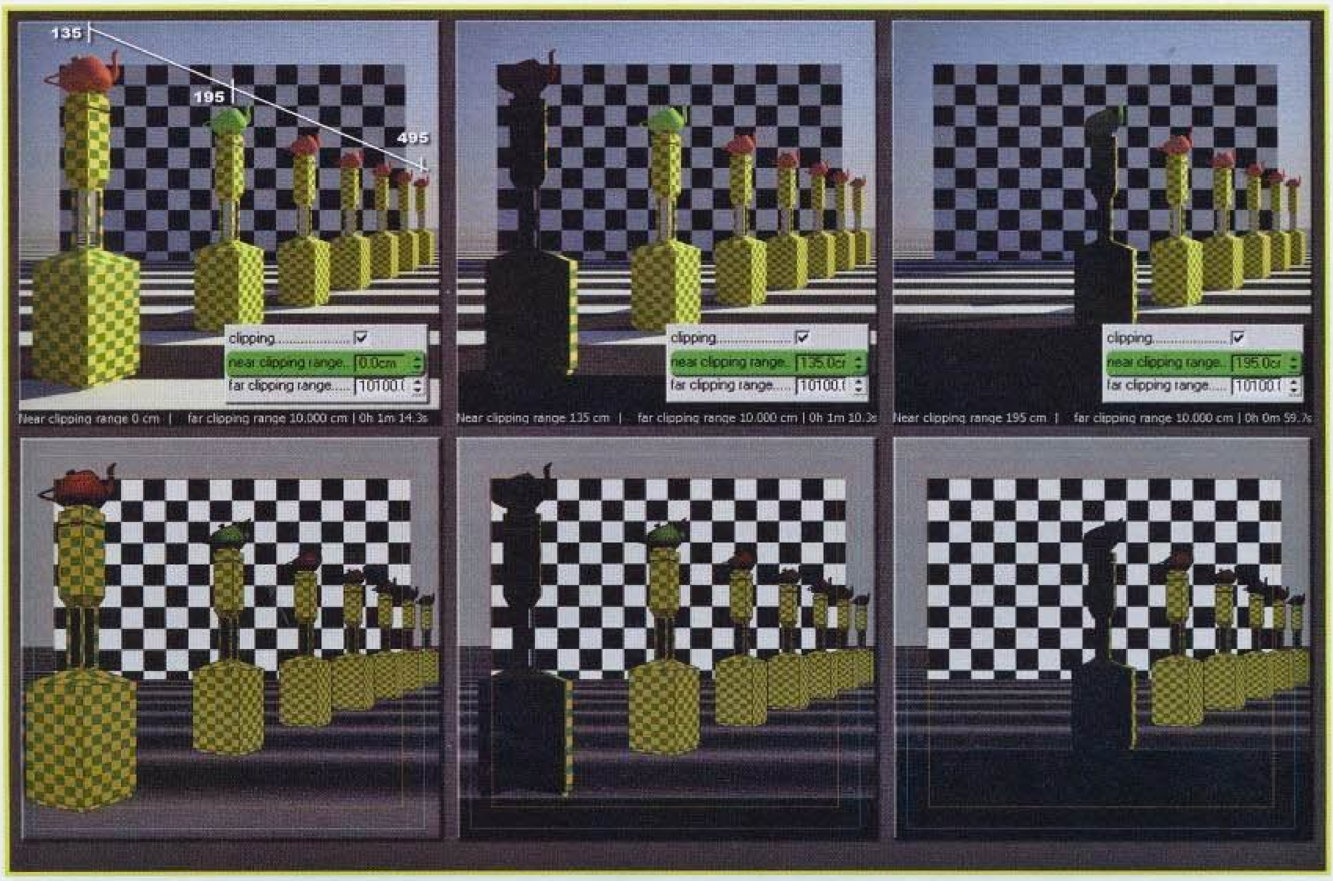
horizon line - by activating this parameter it is possible to observe the appearance of the horizon line in the viewport, inside the ***VRayPhysicalCamera***.



■ **Figure 9.56**
The horizon line visible in viewport.

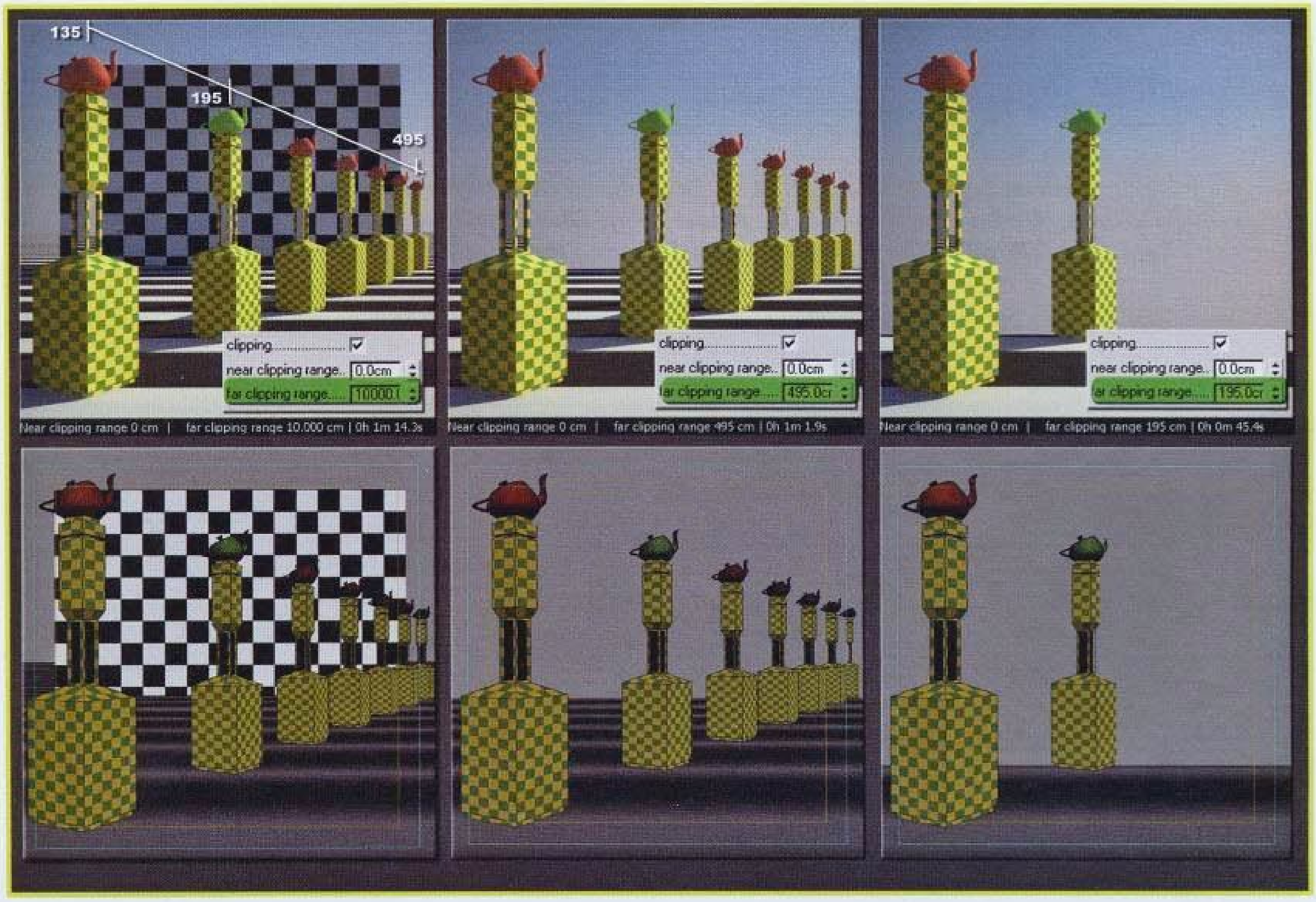
clipping - the activation of this option makes the changing of the ***near clipping range*** and ***far clipping range*** parameters available. Thanks to these two values it is possible to specify at what distance from the camera *VRay* starts to render. The objects closer than the plane defined by the ***near clipping range*** are not visible in the render, like the objects further away than the plane defined by the ***far clipping range***. These planes can be considered as two big knifes cutting the scene in such a way as to make the objects more or less visible.

In the next scene the first column is distant roughly 135 cm from the observation point, the second 195 cm. The last column, instead, 495 cm. By using values of ***near clipping range*** equal to the previous distances, it is possible to specify with high precision the position of the clipping plane. Values lower than the ***near clipping range*** make the objects not visible in the rendering. It is to be noticed how, in viewport too, there is an exact correspondence between what is seen and what is going to be rendered. Vice versa, everything that is over the clipping plane ***far clipping range*** is not going to be computed. In the tests this value corresponds to 10,000 cm and, since no geometry is present over that range, the ***far clipping range*** is non influential.



■ Figure 9.57
It is possible to observe the influence of the ***near clipping range*** both in the viewport and in rendering.

By using different values of ***far clipping range*** one specifies the plane beyond which nothing is rendered anymore.



■ Figure 9.58
It is possible to observe the influence of the ***far clipping range*** parameter both in the viewport and in rendering.

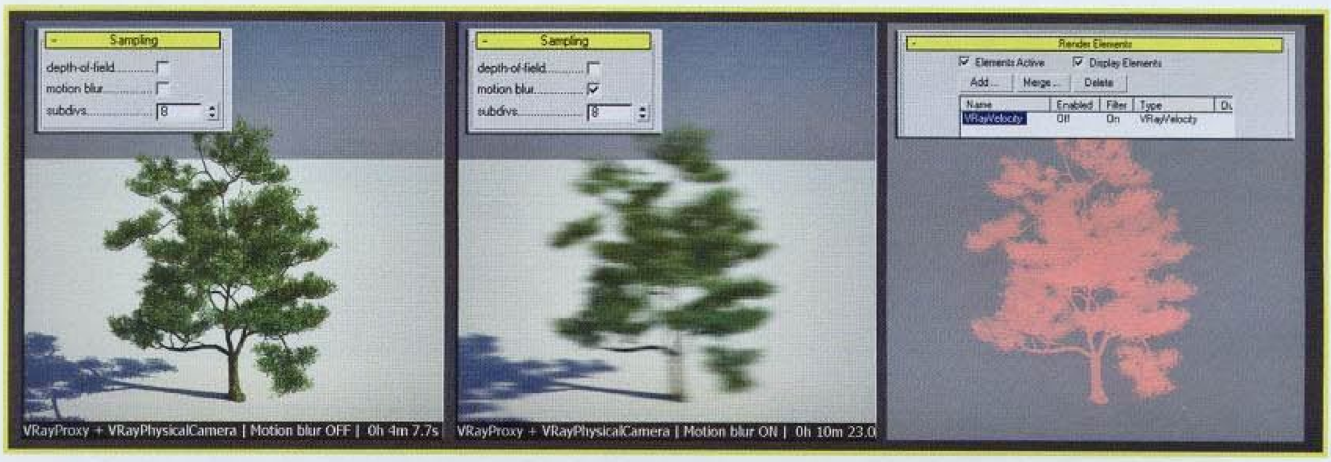
near clipping range - it determines the distance from the clipping plane within which nothing is rendered.

far clipping range - it determines the distance from the clipping plane beyond which nothing is rendered.

Near/far environment plane - it determines the distance from the clipping plane within/beyond which nothing is rendered by using some 3ds Max atmospheric effects.

VRayProxy

Like the ***VRayProxy***, with ***v1.5 FINAL*** the speed of visualization in the viewport has been improved.
It is also possible to generate a correct ***Motion blur*** if animated. Lastly, the ***VRayVelocity*** produces the respective Render Element correctly.



■ **Figure 9.59**
Motion blur generated by a ***VRayProxy***. Specifically a ***VRayPhysicalCam*** has been used. On the right, the ***VRayVelocity*** Render Element.

Textures and Materials

There are many innovations added in ***v1.5 FINAL*** as far as the "textures and shaders" section is concerned. Starting from generic additions, like the ones present in the rollout ***VRay: Global switches***, and going on to new parameters for ***VRayDirt***, ***VRayHDRI***, ***VRayMtl***, ***VRayColor***, ***VRayFastSSS*** and ***VRayEdgesTex***.

Override Exclude - a button activating the appearance of a table with which it is possible to define which objects will be influenced or not by the ***Override Mtl*** function has been added.

■ **Figure 9.60**
Effect of the ***Override Exclude*** tool.

In the test on the left, the scene has been computed with the shaders applied. In the central one, thanks to the activation of the *Override mtl* parameter, the same shader has been associated to all the objects on scene: a neutral shader. In the last test, again with *Override mtl* active, only the liquid has been excluded by using the Exclude/Include tool. This way it has been possible to maintain the original shader, while the remaining geometries have been computed with a grey shader.

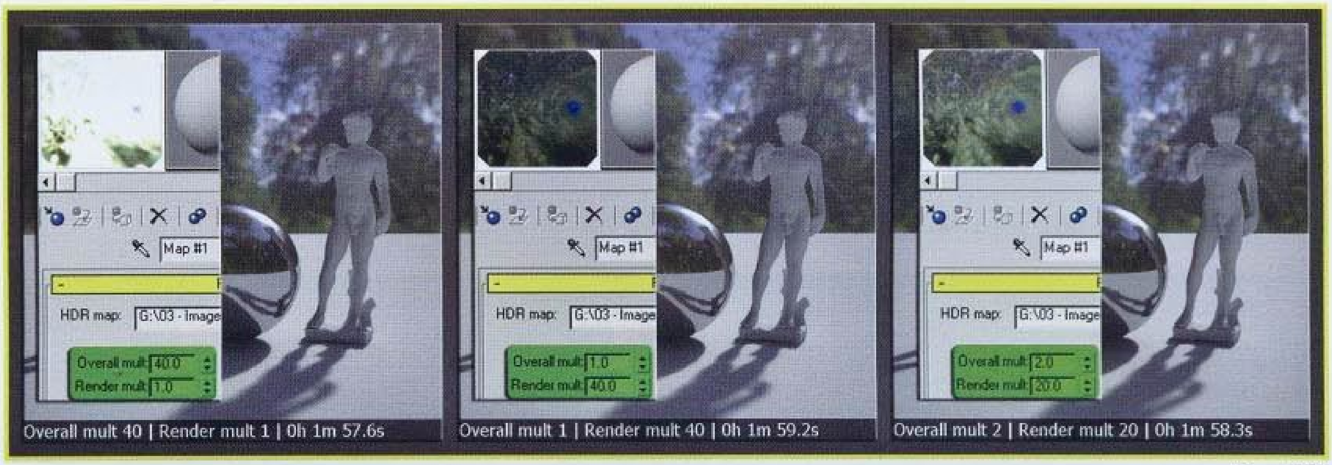
VRayHDRI

Overall multiplier - this parameter controls the luminosity of the HDRI map and influences both the rendering and the Material Editor.

Render multiplier - this parameter controls the luminosity of the HDRI map and influences only the rendering. The luminosity in the Material Editor is not changed.

Often it may happen that one needs high values for the HDRI map during the render. This increase in the luminosity affects the Material Editor as well, obviously, making it practically impossible to visualize the map, because of it being too luminous. This way, for example, the reading of the modification of the vertical or horizontal rotation in the Material Editor is very difficult. By changing the *Render multiplier* parameter only, instead, it is possible to vary the luminosity during the render, keeping the Material Editor luminosity unchanged. Notice how the two multipliers work together. For example, these couples of values produce the same result in the render, but not in the Material Editor:

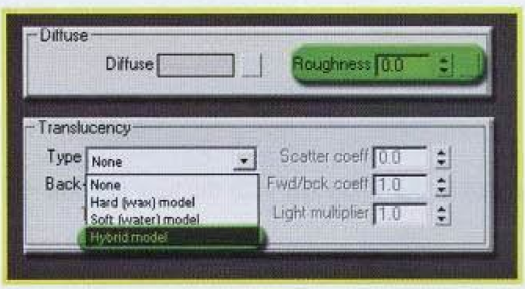
Overall multiplier = 40	*Overall multiplier = 1*	*Overall multiplier = 2*
Render multiplier = 1	*Render multiplier = 40*	*Render multiplier = 20*
Total multiplier = 40	*Total multiplier = 40*	*Total multiplier = 40*



■ **Figure 9.61**
Same results obtained with different values of the ***Overall mult*** and ***Render mult***.

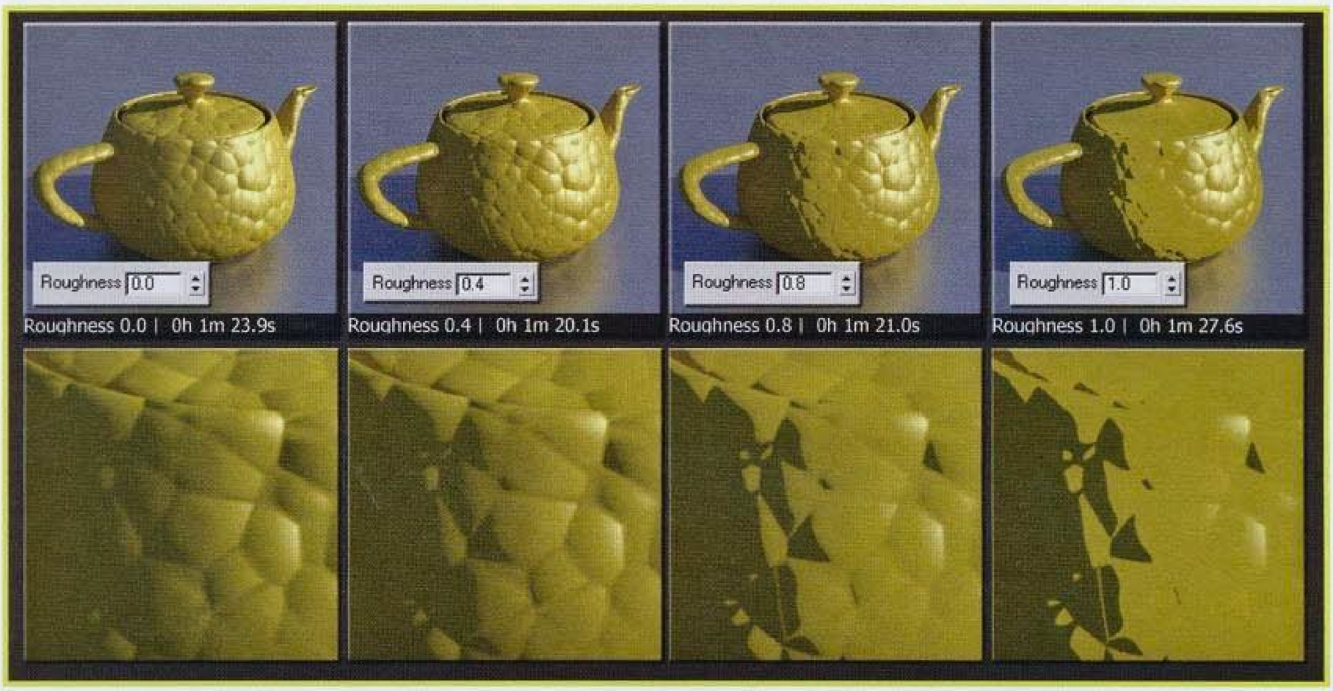
VRayMTL

The *VRayMtl*, has two new parameters in the *v1.5 FINAL* version.



■ **Figure 9.62**
The new parameters of ***VRayMtl***.

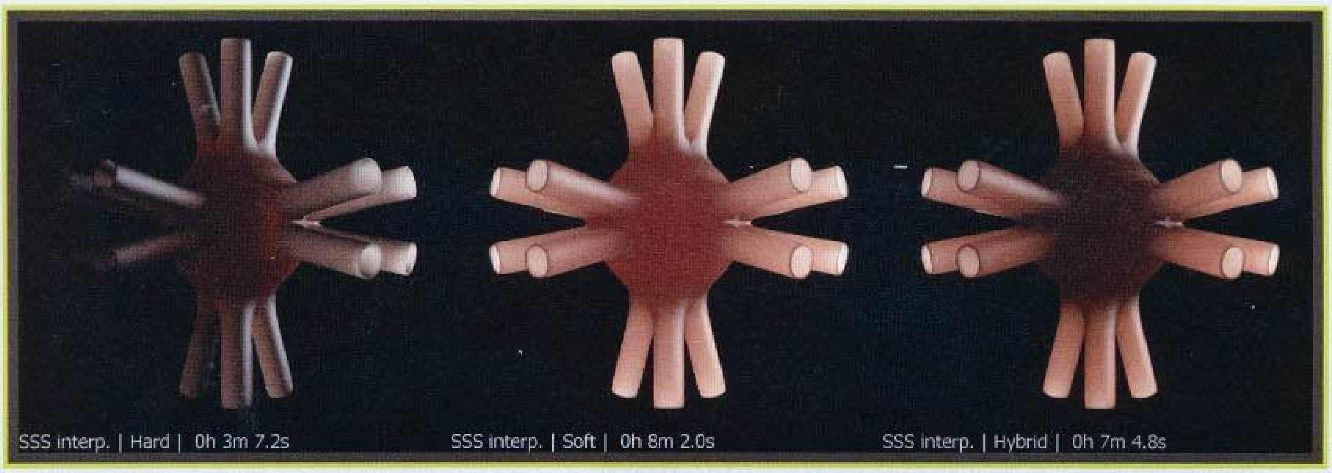
Roughness - this parameter can be used to simulate rough surfaces, like rocks or areas covered by dust. From a technical point of view, this option tends to flatten the shader, by reducing the transition zone from the environmental light to the diffuse one.



■ **Figure 9.63**
Notice the "flat" effect, close to the cartoon, obtained with ***Roughness*** = 1.0.

Hybrid model - three algorithms are available in *VRay* for computing the ***SSS***.

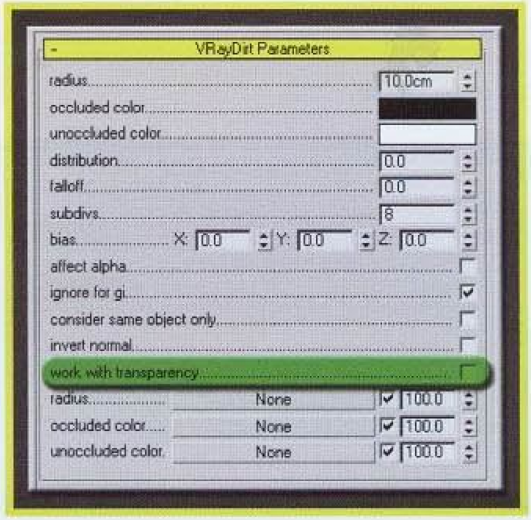
Hard (wax) : useful for simulating materials such as marble.
Soft (water) : model allowing one to improve the compatibility with scenes accomplished with ***v.1.09***.
Hybrid : it is the model yielding the best simulation of the ***SSS***. It is useful for fulfilling materials such as skin, milk, fruit juices and, generically, translucent shaders.



■ **Figure 9.64**
By using the ***Hybrid*** model a mean way between the ***Hard*** and ***Soft*** ones has been obtained. The central part is, in fact, very dark, typical of the ***Hard*** model, while the outermost parts are bright, typical of the ***Soft*** model.

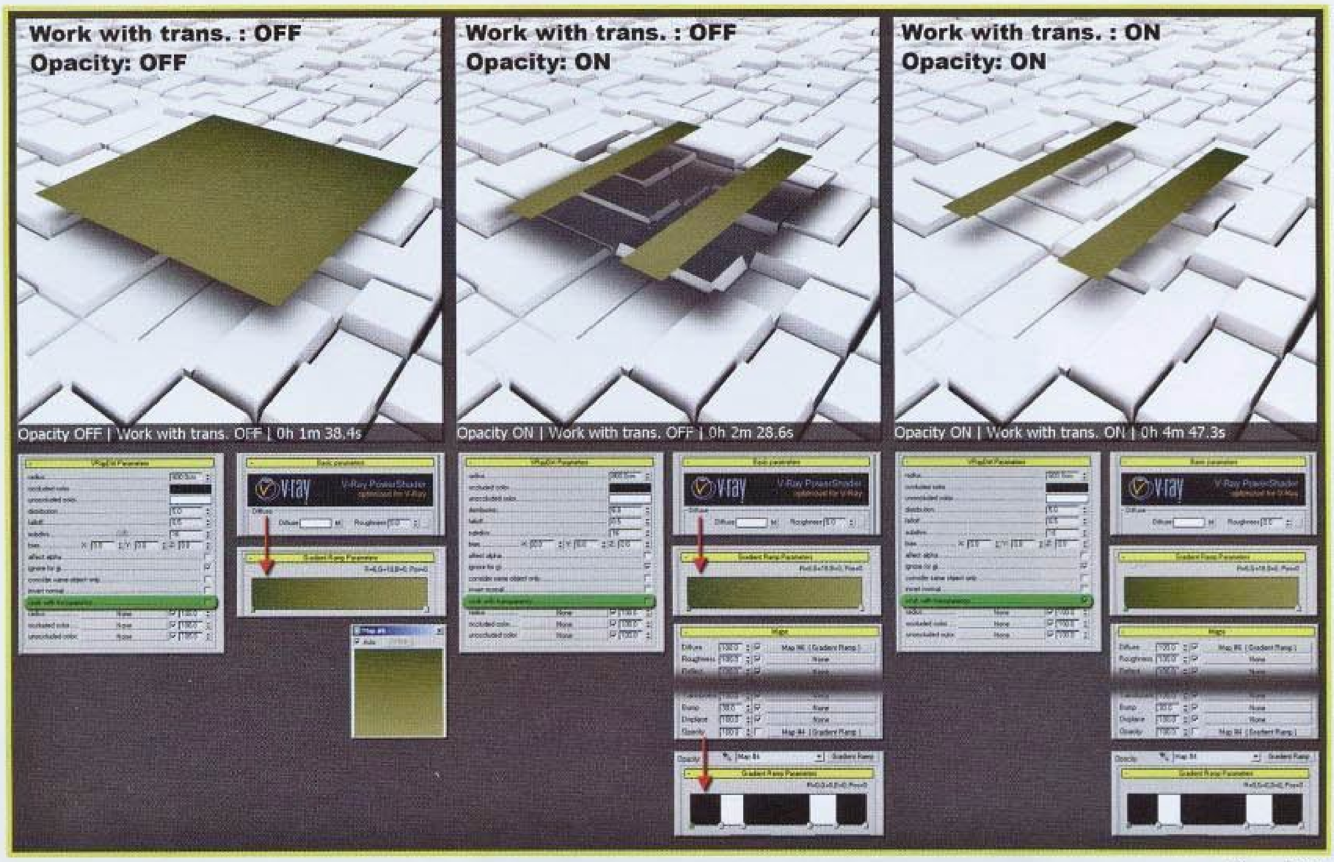
VRayDirt

A new parameter has been added to the ***VRayMtl*** with the new version ***v1.5 FINAL***: ***work with transparency***.



■ Figure 9.65
The new ***VRayDirt*** parameter.

work with transparency - with this parameter, the ***VRayDirt*** map computes the ***Ambient Occlusion*** by taking into account the transparency of objects. This can be useful, for example, if one wants to compute the ***AO*** for objects such as leaves generated through opacity. If it is deactivated, even when there is some transparency, occluded objects are considered solid.

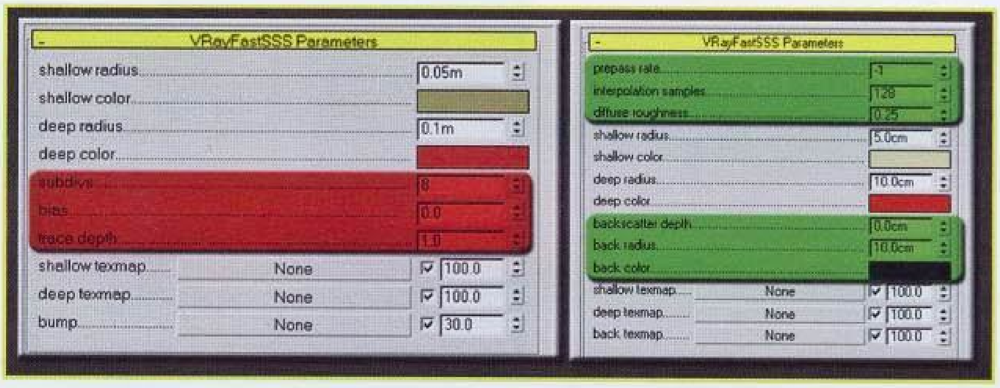


■ Figure 9.66
Use of the ***work with transparency*** option.

Two geometries and two shaders are found in this scene. ***VRayDirt*** is applied to the basement, and a normal ***VRayMtl*** to the plane just above it. In the first image the ***work with transparency*** option is deactivated, while the plane is textured with a simple colored gradient. In the second image the plane is made transparent thanks to an opacity map. But the ***AO*** does not receive this information, continuing to consider the plane completely opaque. Instead, in the last test, ***work with transparency*** has been activated. Now the ***AO*** accomplishes the transparency correctly. Notice how the rendering time has nearly doubled.

VRayFastSSS

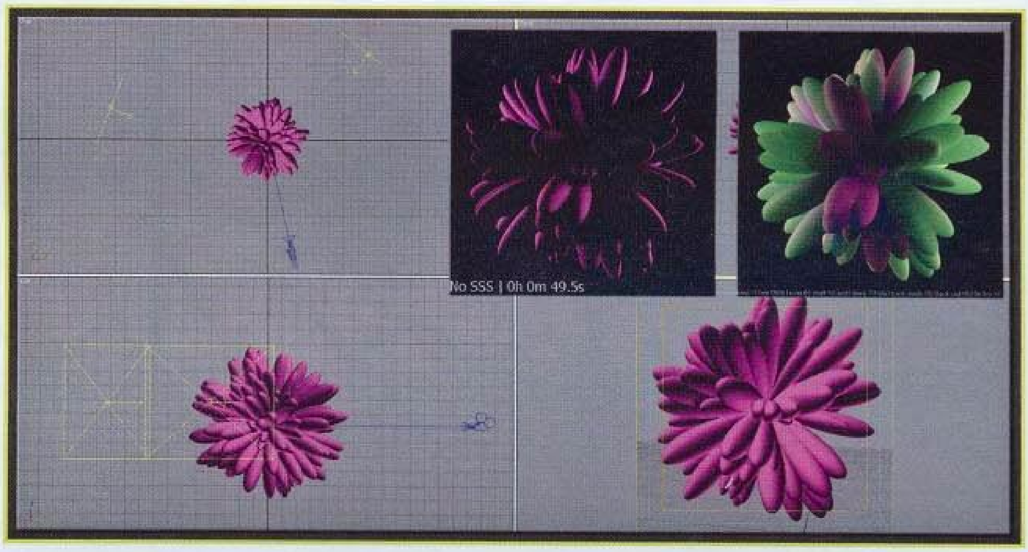
The ***VRayFastSSS*** shader has changed too.



■ **Figure 9.67**
On the left the ***VRayFastSSS*** of ***v1.5 RC3***. On the right ***v1.5 FINAL***. The parameters removed are highlighted in red while the new ones in green.

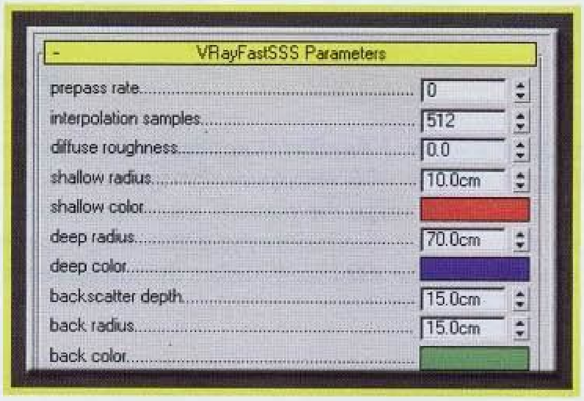
Three parameters of the ***VRayFastSSS*** have been removed from ***v1.5 RC3***, while five have been added in the new ***v1.5 FINAL***. Before ***v1.5 FINAL***, ***VRayFastSSS*** was used to simulate the ***SSS*** through two layers: the ***shallow***, the more superficial one, and the ***deep***, for the transmission of light to the deepest layers. Now a new layer exists, called ***back-scatter***, with the aim of simulating transmission phenomena of the luminous flux of objects in back-light.

The scene examined is made up of an abstract geometry, similar to grapes. Two light sources are placed in the opposite position to the observation point, to produce a translucent effect on the mesh.



■ **Figure 9.68**
The only light sources are ***VRayLights***. Not even the GI is active. This allows one to see the ***SSS*** effect better. On the left the render without the ***SSS***.

prepass rate - ***VRay*** generates the sub-surface scattering by computing the superficial light in critical points on all the geometry, and interpolating them after. This method is very similar to the computing of the GI with the ***IM***. ***Prepass rate*** determines the computing-resolution of this phase. A value = 0 means the ***prepass*** is accomplished with the output resolution. ***Prepass*** = -1 means half of the final resolution. For high-quality renders it is advisable to use the maximum resolution possible.



■ **Figure 9.69**
The initial settings of the ***VRayFastSSS***.

The initial material is a ***VRayFastSSS*** with a high ***prepass rate***. This ensures excellent quality. Also the interpolation value is quite high. The ***diffuse roughness*** parameter has been reset in order to observe the ***SSS*** better. Very different hues have been used for the colors of the ***SSS***, so that they can be seen better within the render. The first layer, the ***shallow***, is about 10 cm deep. One must be aware that the diameter of the geometry is about 1 meter. Instead the secondary layer acts up to a depth of 70 cm. Also the ***backscatter*** effect is active, which intervenes on areas exposed to direct light, in this case the green parts.

■ **Figure 9.70**
Effects on quality and rendering time generated by the changing of the ***prepass rate*** parameter.

With ***prepass rate*** = -3 one can see all three ***SSS*** levels. The green color is present only in the outermost parts, because that is where the light of the two ***VRayLights*** hits the back of the geometry. The blue-purple parts instead, are not exposed to direct light and return the colors of the ***shallow*** and ***deep*** layers. As the number of ***prepass rates*** increases, the geometry becomes, in the central areas, darker and darker. This happens because as ***prepasses*** increase it would be wise to increase also the ***interpolation sample*** parameter. If there are not enough samples the render blackens. Notice how rendering time has tripled.

interpolation sample - this parameter determines the number of samples used for ***SSS*** interpolation, calculated in the ***prepass*** phase. For very translucent surfaces, this parameter should be very high.

■ **Figure 9.71**
It is evident that increasing ***interpolation samples*** causes a general increase in ***SSS*** quality.

The increasing of the ***interpolation samples*** parameter immediately improves quality. The zones which were previously completely black, more difficult to illuminate with translucency, are no longer so.

diffuse roughness - like the ***VRayMtl***, it is a parameter which allows the "flattening" of the shader. High values increase translucency.

■ **Figure 9.72**
In this example, changing the ***diffuse roughness*** parameter slightly modifies the appearance of the ***SSS***.

shallow radius - this parameter determines, in units referred to the scene, the transmission of light in the first layer. Low values lead to less ***SSS*** and faster renders. High values accentuate the ***SSS***. Increasing the ***SSS*** depth requires an increase in ***interpolation samples***. For practical examples, observe those created for the ***VRayFastSSS*** of ***v1.5 RC3***.

shallow color - this parameter represents superficial ***SSS*** color. Notice that the color inserted into this channel acts as a filter for the remaining colors. If ***shallow color*** is white, neither ***deep*** or ***backscattering*** are taken into account. This layer can also be controlled by a texture. For practical examples observe those fulfilled for the ***VRayFastSSS*** of the ***v1.5 RC3***.

deep radius - this determines the depth of light transmission in the second layer of the ***SSS***. Usually one uses settings higher than those of the ***shallow radius***. For practical examples see those for ***v1.5 RC3***.

deep color - this parameter controls the ***deep*** layer color. This color is used as a filter for ***back-scattering***. If the ***deep color*** is white, there can be no ***back-scattering*** effect.

backscattering depth - this parameter represents the radius of action of the ***back-scattering*** effect.

■ **Figure 9.73**
Increase in the ***backscattering*** effect.

In the first image, although ***backscattering*** has a color, it is not present in the render. This happens because the depth of action is null. By increasing the ***backscatter depth*** parameter, one starts to see a certain illumination along the edges of the geometry. Notice that the radius has been appropriately reset to zero. This has been an attempt to concentrate the action of the ***SSS*** within the threshold defined by the depth only. Rendering time has increased considerably.

Back radius - this parameter represents the range of action of the ***back-scattering*** effect. High values increase the diffusion of the effect, but make the render slower. For better quality it is advisable to use high settings of ***interpolation samples***.

Figure 9.74
At equal ***depth***, by increasing the range, also the depth of action of ***back-scattering*** increases, as well as rendering time.

VRayEdgesTex

The ***VRayEdgesTex*** has an extremely interesting feature: the possibility of "simulating" rounded edges on hard-edged geometrical objects. It is similar to bump mapping, which involves the texture simulating "at render time" the roughness which the geometry does not actually possess. In this case, the same thing takes place, except that the simulation regards rounded edges.
To obtain this effect one must insert the ***VRayEdgesTex*** map in the Bump channel of the material and use ***World units*** to manage the amount of smoothness. The ***Pixel*** option in this case is ignored and is of little use.

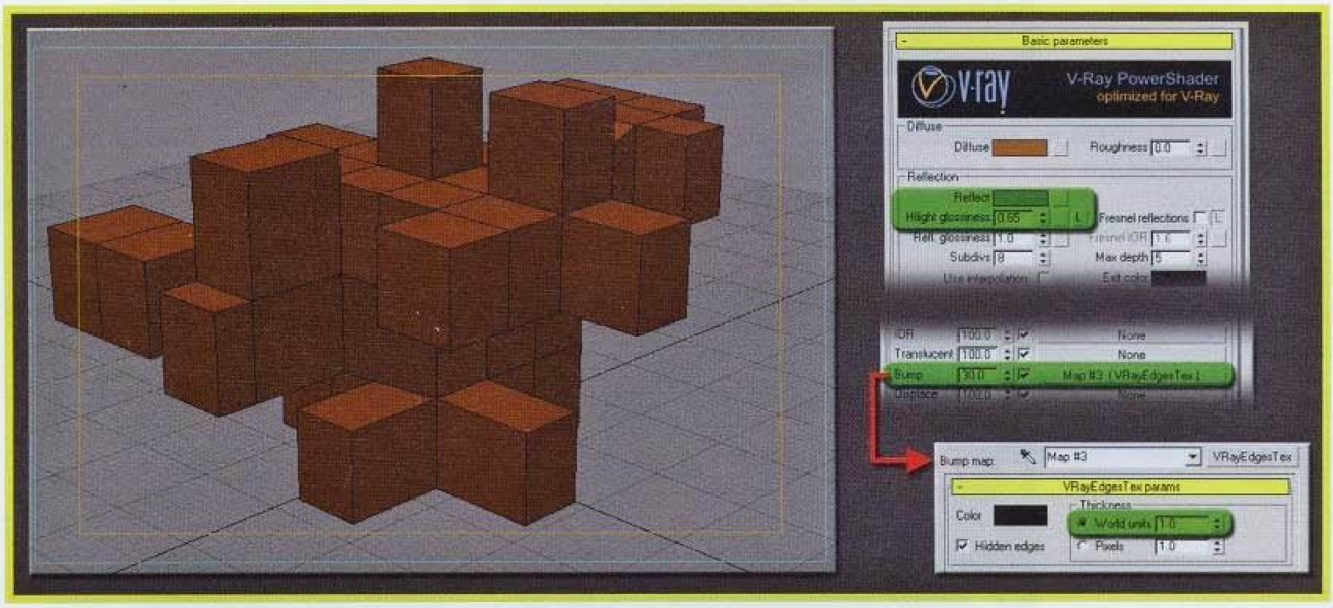
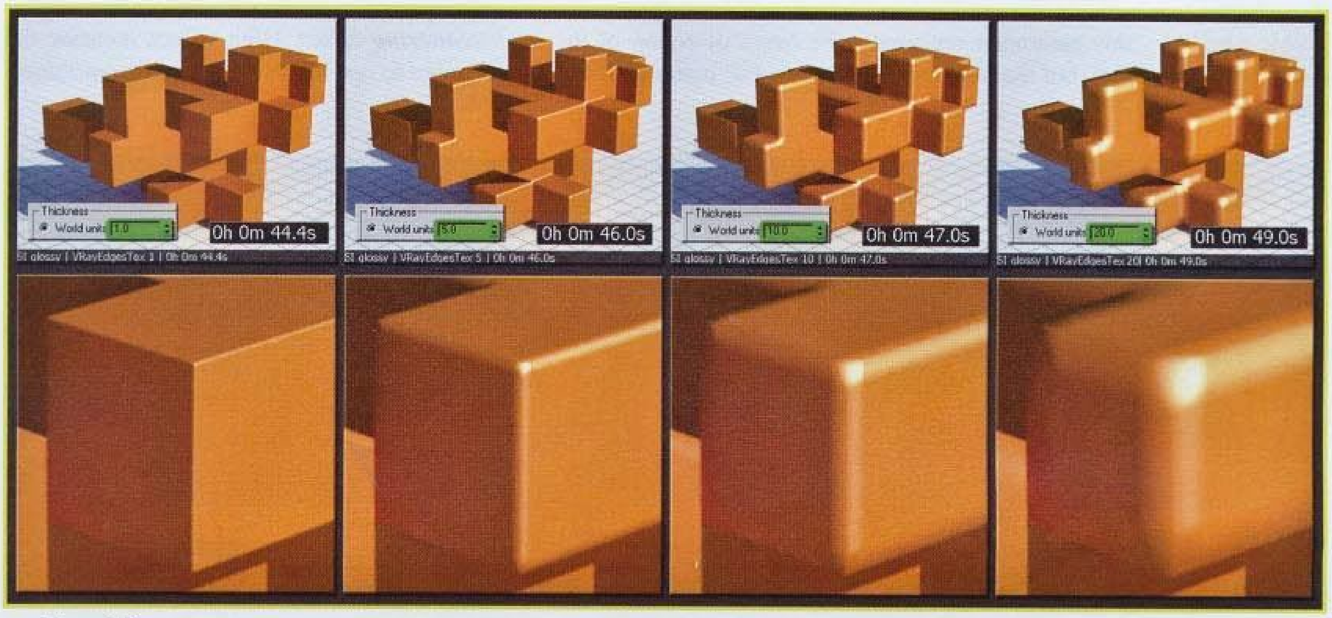


Figure 9.75
The geometry is a simple mesh with 90° edges, without geometrical smoothing. The material is a ***VRayMtl*** with non-glossy ***highlights*** active. In the Bump channel a ***VRayEdgesTex*** has been inserted with ***World units*** set to 1.0.

The result obtained by changing the ***World units*** parameter, is an increase of the artificial rounding of edges generated during rendering. Calculation time, as the range increases, undergoes extremely small variations. Beyond a certain threshold, in this case between 10 and 20 units, the artificial smoothing shows its true colors, giving an unsatisfactory result. With ***World Units*** < 10 the effect is very realistic but with wide ranges loses quality.

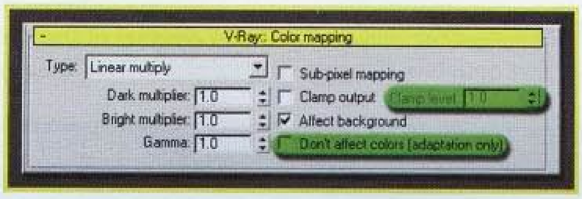


■ **Figure 9.76**
Edge-smoothing generated by the ***VRayEdgesTex*** map.

Color mapping

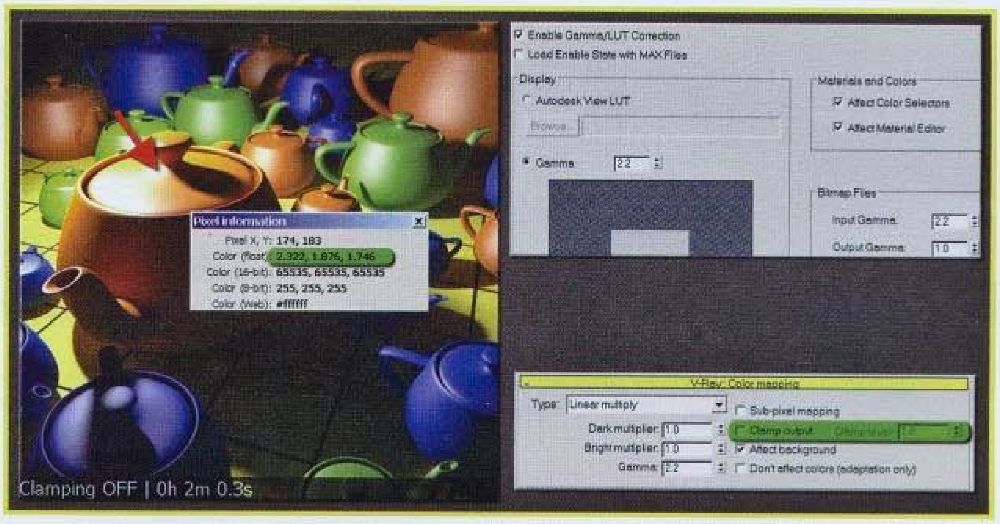
The ***VRay: Color mapping*** rollout adds, from ***v1.5 FINAL*** onwards, two new options: ***Clamp level*** and ***Don't affect colors (adaptation only)***.

■ **Figure 9.77**
The two new parameters of the ***VRay: Color mapping*** rollout.



Clamp level - if ***Clamp output*** is active, this parameter specifies the amount of color clamping. When a luminous flux hits a colored geometry, if the source is too intense the render can generate colors which are above float = [1,1,1]. In this situation, "on monitor" the image appears overexposed, burnt. By measuring with the ***Pixel information*** tool it is possible to find the real values of the pixel.

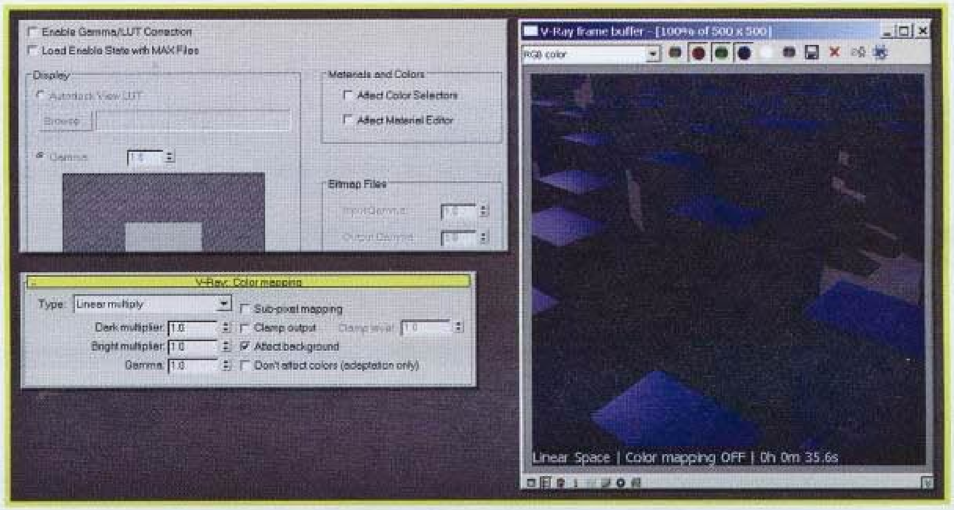
■ **Figure 9.78**
In this image the highest value is roughly float = [2.3, 1.8, 1.7].



In the previous image, in the areas of maximum illumination one has reached a float value of = [2.3, 1.8, 1.7]. By activating the ***Clamp output*** parameter and with ***Clamp level*** = 1.0 the result is obviously float = [1.0,1.0,1.0], because ***Vray*** clips the colors above 1.0. By changing ***Clamp level*** to = 1.5, one obtains a maximum float = [1.5, 1.5, 1.5].

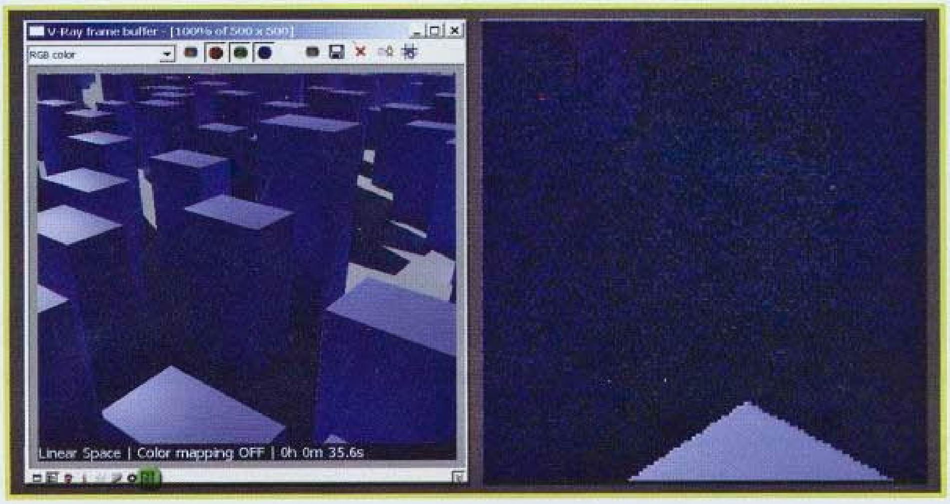
■ **Figure 9.79**
The values of maximum illumination are "blocked" by the ***Clamp level*** parameter.

Don't affect colors (adaptation only) - with this parameter active, in the final render there will be no correction carried out by ***Color mapping*** and the image will be linear, with ***Gamma*** =2.2. The difference is that the calculation of effects, like the GI, is computed as if ***Color mapping*** were active. This is useful if one wants to obtain an image in linear space, knowing that the render will be modified later by applying a few color corrections, but with the advantages of a ***Color mapping*** used during the calculation phase.
In the next example 3ds Max is set to linear space, just like ***VRay Color mapping***. The image in the ***VFB*** is dark.



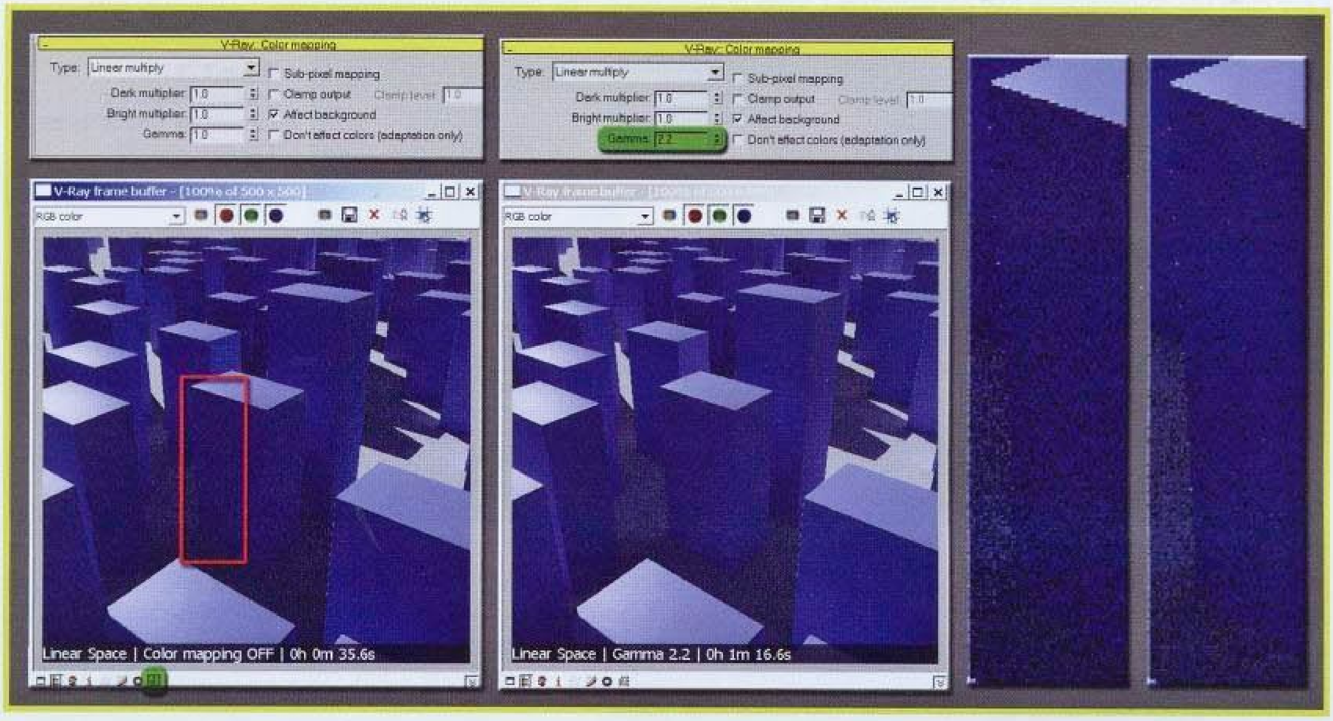
■ **Figure 9.80**
Image in linear space. All the parameters are default.

Now activate the ***sRGB*** option in the ***VFB***.



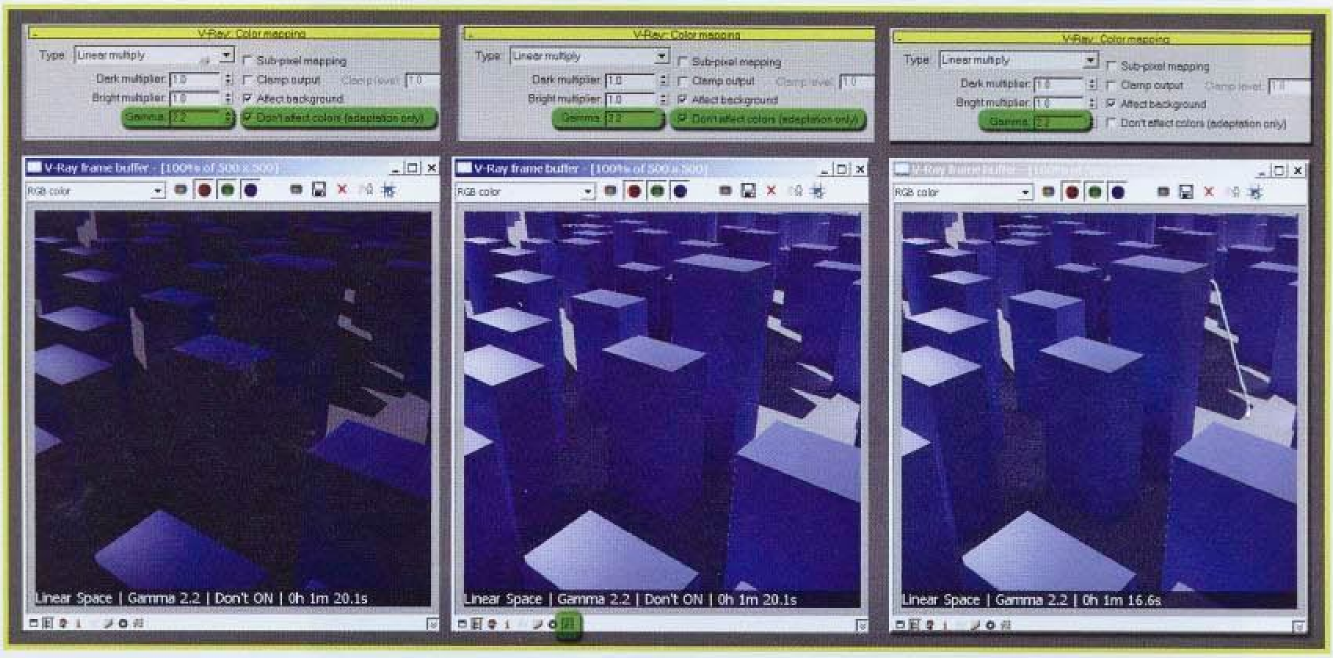
■ **Figure 9.81**
Image in linear space but visualized in ***sRGB*** linear space. The render has been generated with the ***Brute force + Brute force*** method without ***antialiasing***. This has lead to a high noise level.

The image is correct and a certain level of noise can be seen in the darker areas. At this point, deactivate the ***sRGB*** button and modify ***Gamma*** = 2.2. Observe the render in the ***VFB*** and compare it with the one in linear space.



■ **Figure 9.82**
Lower noise due to activating the ***Gamma*** in the render calculation. Notice the increase in rendering time caused by the greater energy in the scene.

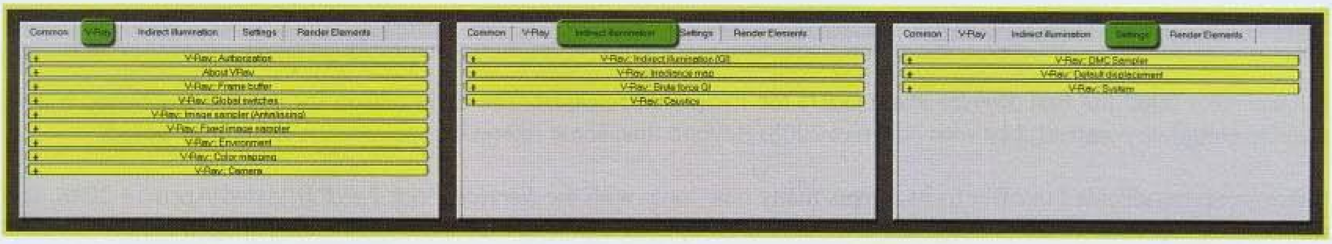
By acting at Gamma level and not during post-production (as would happen if one activated the ***sRGB*** button), there is more energy in the scene because of the greater brightness. Consequently also the GI is changed and the areas which were previously darker, and therefore hardly affected by the GI, are no longer so and are sampled better. The result is a lower amount of noise. In this case, the render generated is however not in linear space, but has a certain amount of correction.
It is now time to activate the ***Don't affect colors (adaptation only)*** option. The render visualized in the ***VFB*** is dark as if ***Gamma*** = 1.0. By activating the ***sRGB*** button, however, we have a pleasant surprise: noise has decreased, as if ***Gamma*** were equal to 2.2.



■ **Figure 9.83**
The image is in linear space, but has the same noise obtainable by changing the ***Gamma***.

Renderer

Before ***v1.5 FINAL*** the Renderer panel contained all the rollouts for the management of ***VRay*** in one panel only. Now it has been divided into three different areas: ***VRay***, ***Indirect Illumination*** and ***Settings***.



■ **Figure 9.84**
For an improved management of all the VRay rollouts, the Renderer panel has been subdivided into 3 panels.

The ***VRay*** panel contains most of the rollouts, whereas the ***Indirect Illumination*** section represents the sections which are dedicated exclusively to GI management, therefore the ***Irradiance Map***, ***Brute Force***, ***Photon Map*** and ***Light Cache***. The ***Settings*** panel contains the rollouts dedicated to the more advanced options in ***VRay***, such as the ***DMC Sampler***.

INTRODUCTION

In December 2007 the first ***Service Pack (SP1)*** for *VRay* was released. The main reason for it being published was to provide complete compatibility with 3ds max 2008. Beyond this, some interesting parameters have been added.

Chaosgroup has decided to offer to its clients many new tools with the ***Service Pack 2 (SP2)***, dated April 11 2008, and this actually represents a deeper upgrade compared to ***SP1***. Here below are the main added and changed tools.

SP1 - NEW FEATURES	VARIATIONS AND ADDITIONS
COMPATIBILITY	
Support	Complete support for ***3ds max 2008***.
TEXTURES & SHADERS	
Anisotropy	Added the possibility of creating anisotropic shaders through maps and textures.
GEOMETRIES	
VRayProxy	Added the ***Scale*** parameter.
VRayFur	Added the ***Taper*** parameter.
RENDER ENGINE	
Sampling	***Schlick*** is now the only ***path sampler*** supported by *VRay*.
RENDER OUTPUT	
Clamp output	The clamping continues to affect the image, even if the ***Don't affect colors*** parameter is active.
Color mapping	Now it does not affect the Background when seen through matte objects if ***Affect background*** is active.

SP2 - NEW FEATURES	VARATIONS AND ADDITIONS
COMPATIBILITY	
Support	Complete support for 3ds max 2009 and 3ds Max Design 2009.
TEXTURES & SHADERS	
VRayMtl	Added the ***Soften*** parameter.
VRaySimbiontMtl	Direct support from *VRay* for the DarkTree shader.
GEOMETRIES	
VRayProxy	Support for the use of ***animated*** geometries.
VRayProxy	New ***point*** visualization mode.

LIGHTS	
VRayIES	New kind of light allowing the use of photometric files .ies.
VRayLights	Now it is possible to use textures similarly to the ***dome light***.
VRayMapShadows	New kind of shadow aimed at reducing flickering in animations achieved with ***VRayFur***.
CAMERAS	
VRayPhysCam	Added new presets for the white balance. By default it is ***D65***.
RENDER ELEMENTS	
VRay_Illuminance	Useful channel for the analysis of luminous intensity.
VRay_BumpNormals	Useful channel for extracting the Normal bump channel.
VRay_ExtraTex	Useful channel for extracting any texture applied to all the geometries in the scene.
VRay_SampleRate	Useful channel for observing the ***image sampling rate*** (antialiasing samples).
OTHER	
Profile	Added the preset ***MAX.vray*** used to set *VRay* as the default Render.

SP1 PARAMETERS

Textures and Shaders

VRayMtl

With the ***v1.5 FINAL SP1***, the possibility of using textures for the management of the anisotropy has been added in the ***VRayMtl.*** Bear in mind that this particular effect can be useful for the simulation of typical shaders of anisotropic materials, like metallic objects.

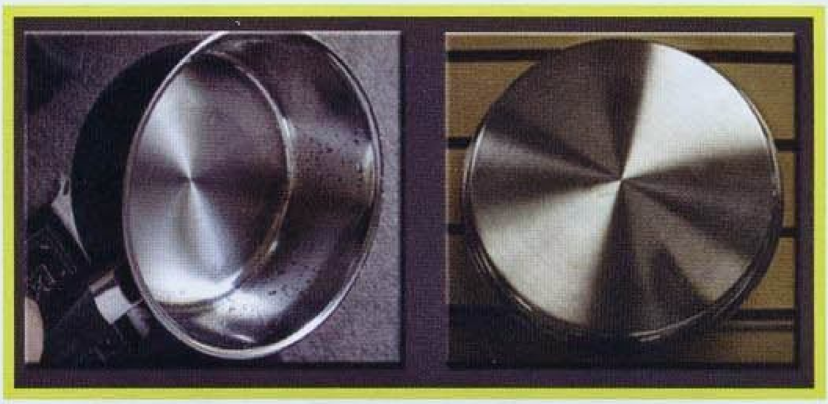
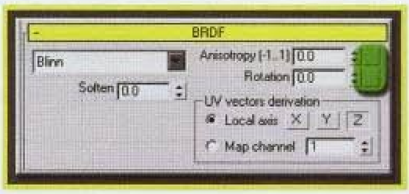


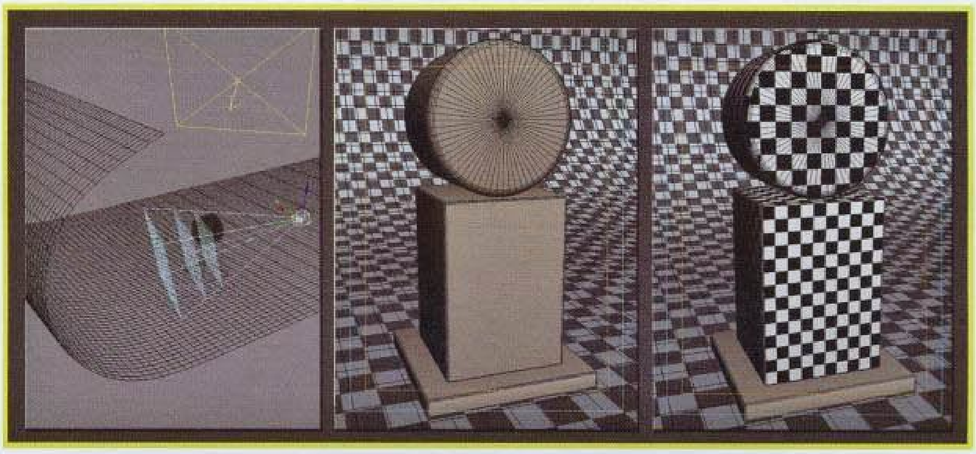
Figure 9.85
Real examples of anisotropy.

The parameters allowing one to use images for creating this effect are located in the ***VRayMtl***, more precisely in the ***BRDF*** rollout, next to ***Anisotropy*** and the ***Rotation***.



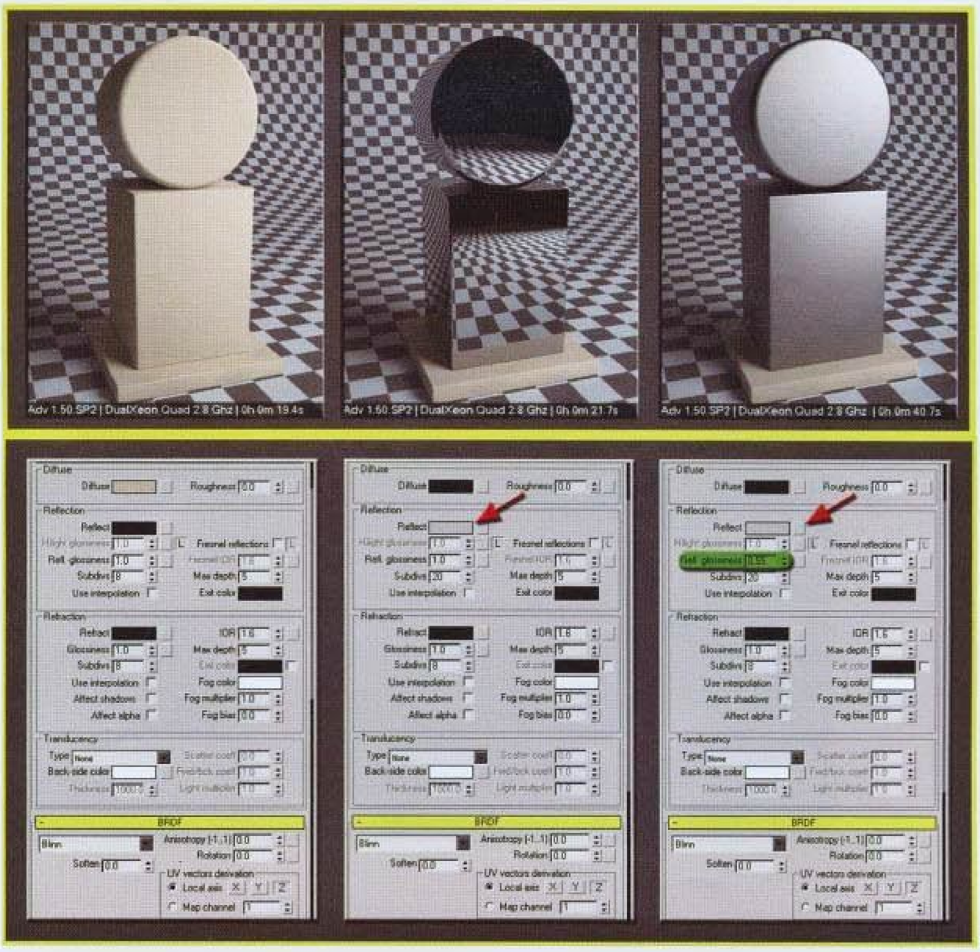
■ **Figure 9.86**
The two slots in which it is possible to use the maps for the management of the anisotropy have been highlighted.

A simple scene is used for the demonstration. A floor and two primitives illuminated by a ***VRayLight Plane***. The whole of this is managed in the GI with an ***IM+LC*** system. The geometries have been chosen for two reasons. The cylinder is used to produce an effect similar to the behaviour one would witness if using a pot, instead the box allows one to observe the effects of anisotropies on a flat surface. Both the objects have UVW coordinates applied and managed with an UVW Mapping modifier.



■ **Figure 9.87**
The scene used for the anisotropy test.

Here below the scene rendered with a default material, a chromium-plated material and a glazed material.

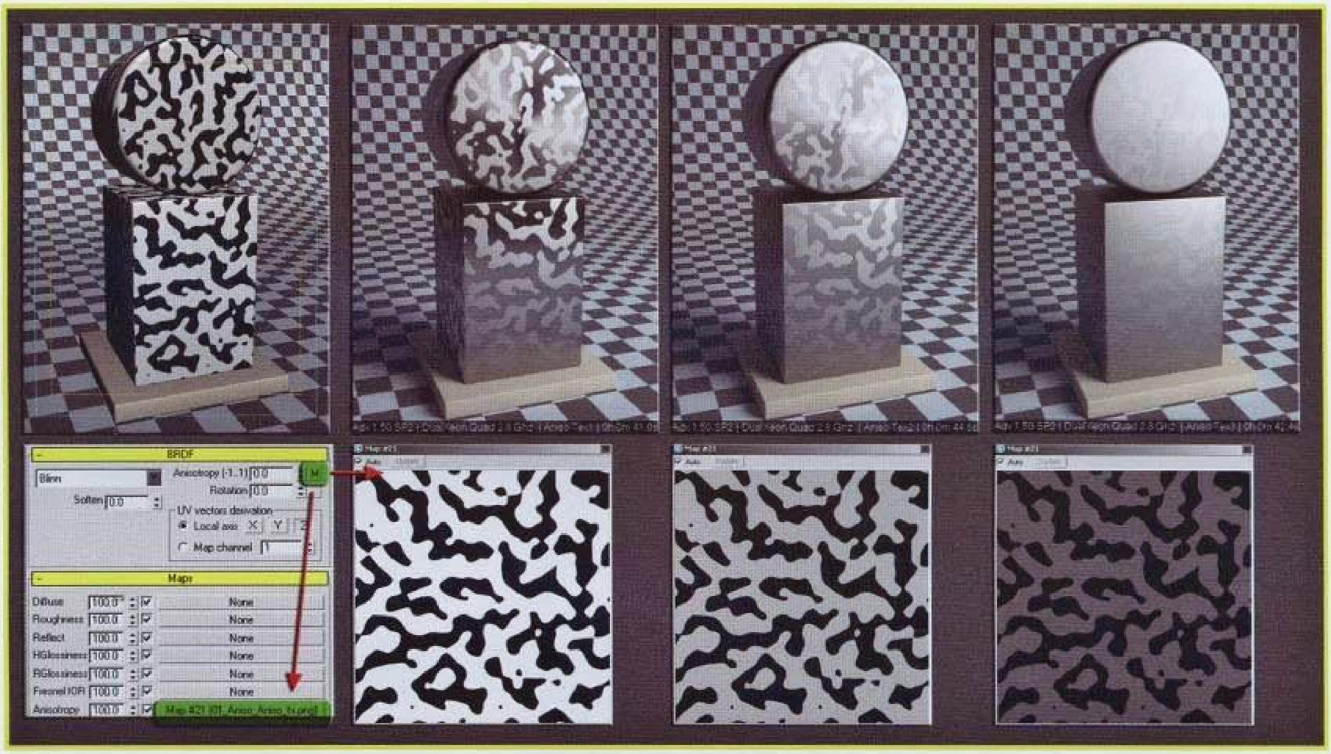


■ **Figure 9.88**
Rendering in neutral mode, steel and glazed steel.

Now the ***Anisotropy*** parameter must be modified.

■ **Figure 9.89**
Variation of the ***Anisotropy*** parameter. Notice how the shape of the ***highlights*** changes.

Now some textures in grey scale are applied.

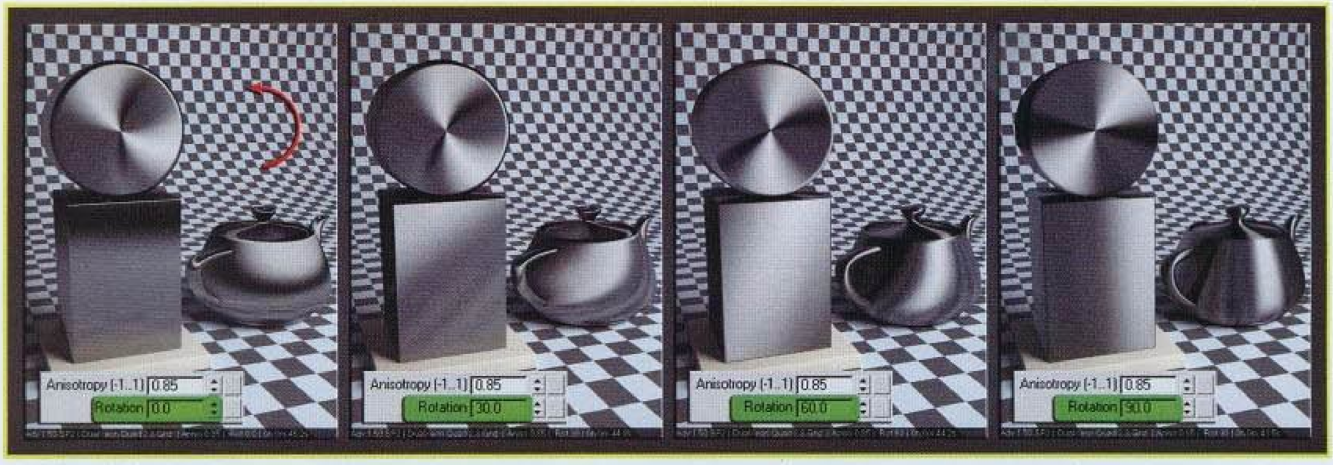


■ **Figure 9.90**
Variation of the ***Anisotropy*** parameter through the use of a Bitmap.

The texture which has been used is a simple Bitmap in black and white. Therefore this map controls the anisotropy, more or less like one would for the specularity or the bump channel. In the black spots on the texture no anisotropy is to be applied, whereas in the white ones there will be maximum anisotropy.

As for the ***Rotation*** parameter, the basic idea is very similar to the previous parameter. With the numerical spinner one can modify the rotation of the highlights. Instead, by using a black and white map it is always possible to define the rotation of the highlight, but, as is to be seen soon, with a much higher precision and customization. Concluding, by using these two maps the user can decide exactly where to locate and how to represent the anisotropic effect.

In the next test a spherical geometry is added. This way it is easier to understand the rotation of the highlights.



■ **Figure 9.91**
Anti-clockwise rotation of the highlights generated by the change of the ***Rotation*** parameter.

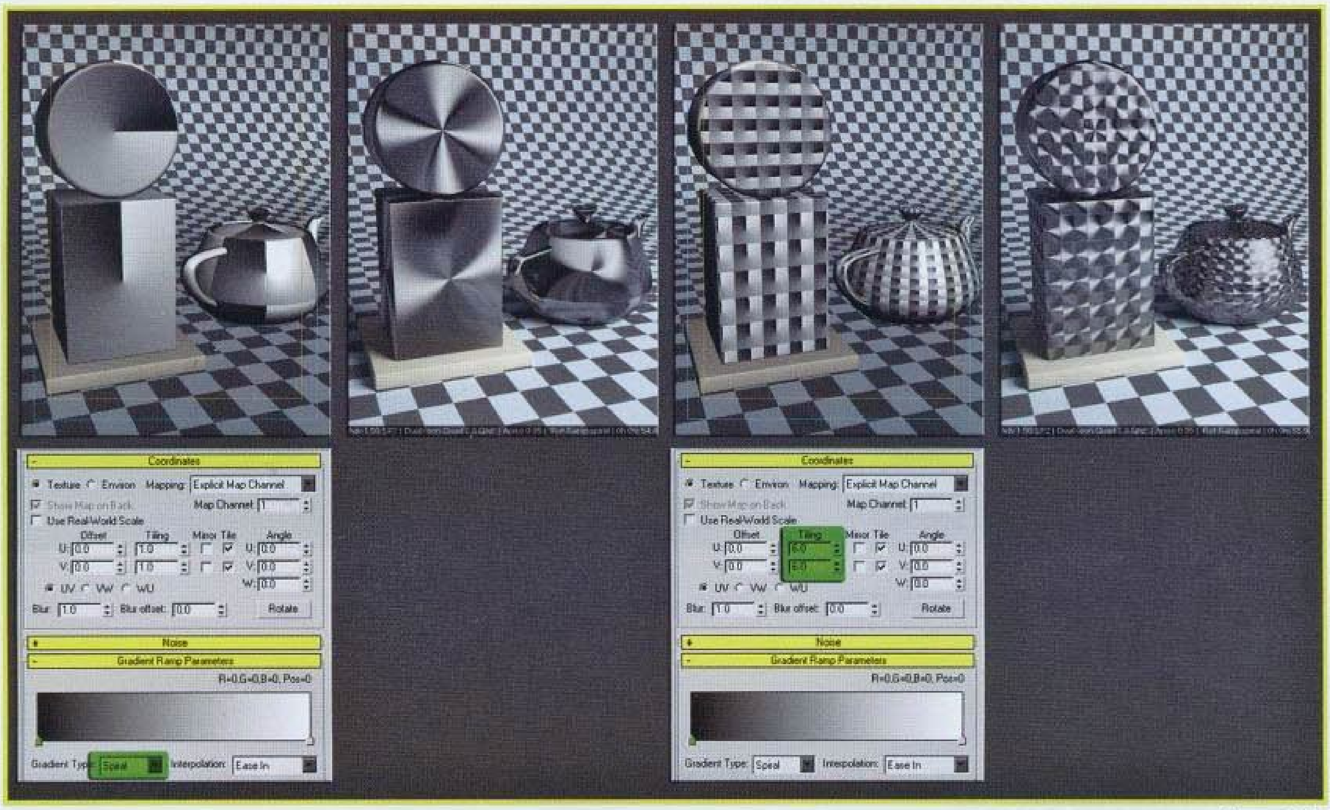
At this point one must not use the spinner for rotation anymore, but a uniform color, generated by the ***Gradient Ramp*** procedural map. Thanks to the color it is possible to define the degree of rotation of the highlights.

Color 1 = RGB[0,0,0]	---> Rot 0°
Color 2 = RGB[21,21,21]	---> Rot 30°
Color 3 = RGB[42,42,42]	---> Rot 60°
Color 4 = RGB[64,64,64]	---> Rot 90°
Color 5 = RGB[128,128,128]	---> Rot 180°
Color 6 = RGB[192,192,192]	---> Rot 270°
Color 7 = RGB[255,255,255]	---> Rot 360°

■ **Figure 9.92**
Anti-clockwise rotation of highlights generated by the change of the ***Rotation*** parameter by using a grey-scale texture.

As one can see, the renders generated with the spinner and the texture are nearly the same. At this point one might ask: "Why should I make things more complicated by using textures when it is possible to obtain the same result by using another method?" The answer relies in the flexibility and the infinite possibilities that the texture can offer.

Now change the gradient from simple solid color to spiral. Then modify the Tiling of the texture in order to multiply its amount.

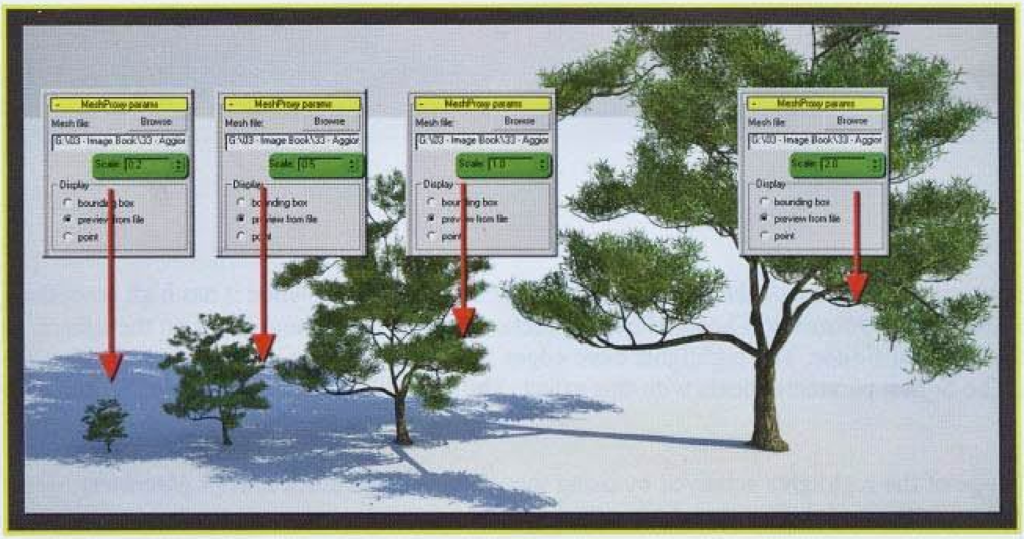


■ **Figure 9.93**
Generation of anisotropic effects on flat surfaces.

Now it can be observed how a very realistic anisotropic effect has been reproduced on the flat surface of the box. This happens because the highlights are continuously rotated thanks to the gradient. By multiplying it, the effect tends to be similar to what happens on some steel-brushed panels. This is just a little example of the potential of this new tool.

VRayProxy

Scale - with this parameter it is possible to scale the ***VRayProxy*** object in a parametric way.



■ **Figure 9.94**
VRayProxy object, copied four times as "copied" and scaled with the ***Scale*** parameter.

VRayFur

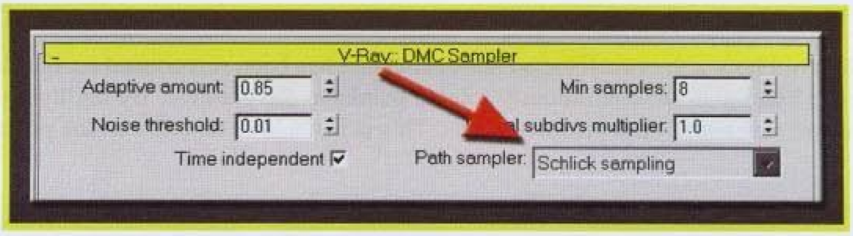
Taper - with this parameter it is possible to taper the top of the filaments. It is enough to modify the parameter up to 1.0 for having a complete tapering, with sharp ends.

■ **Figure 9.95**
By increasing the tapering, rendering time increases as well.

Sampling

As one can see from the screenshot, now the only available ***path sampler*** is ***Schlick sampling***. The pull-down menu is darkened and no longer selectable.

■ **Figure 9.96**
The improvements provided by the ***Schlick sampling*** have allowed to remove the two previous ***path samplers***.



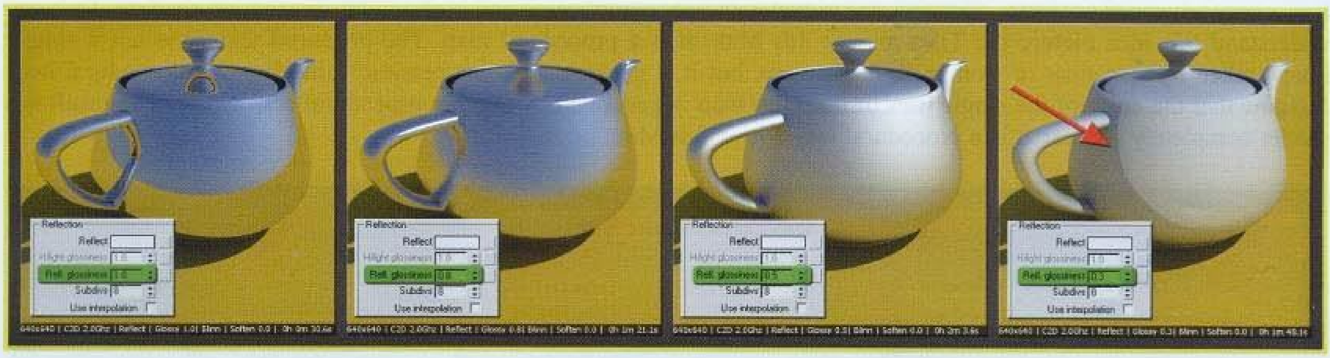
SP2 PARAMETERS

Textures and Materials

VRayMtl

Soften - a new parameter allowing one to manage the transition of highlights in a material when it has high smoothed-reflection features, has been added to the ***VRayMtl***. The maximum reflection point is a luminous point on the sphere, as one can see in the slots of the Material Editor. The highlights have edges which tend to diminish in brightness as one gets further from the centre. The ***Soften*** parameter deals with this effect. The same parameter is found in the Standard material of 3ds Max.

We start by observing the change of the highlights achieved by using increasingly low values of ***Refl.glossiness***, down to a value ***Refl.glossiness*** = 0.3, very low and quite unusual parameter in production.



■ **Figure 9.97**
Drastic reduction of the glossy. The shader ***Blinn*** has been used.

By using very low ***glossy*** parameters, it may happen that a very evident dividing line appears in the area of transition from the highlight to the surface in shadow. To delete it, it is necessary to use the ***soften*** parameter.

■ **Figure 9.98**
Clear improvement in the quality of the highlight.

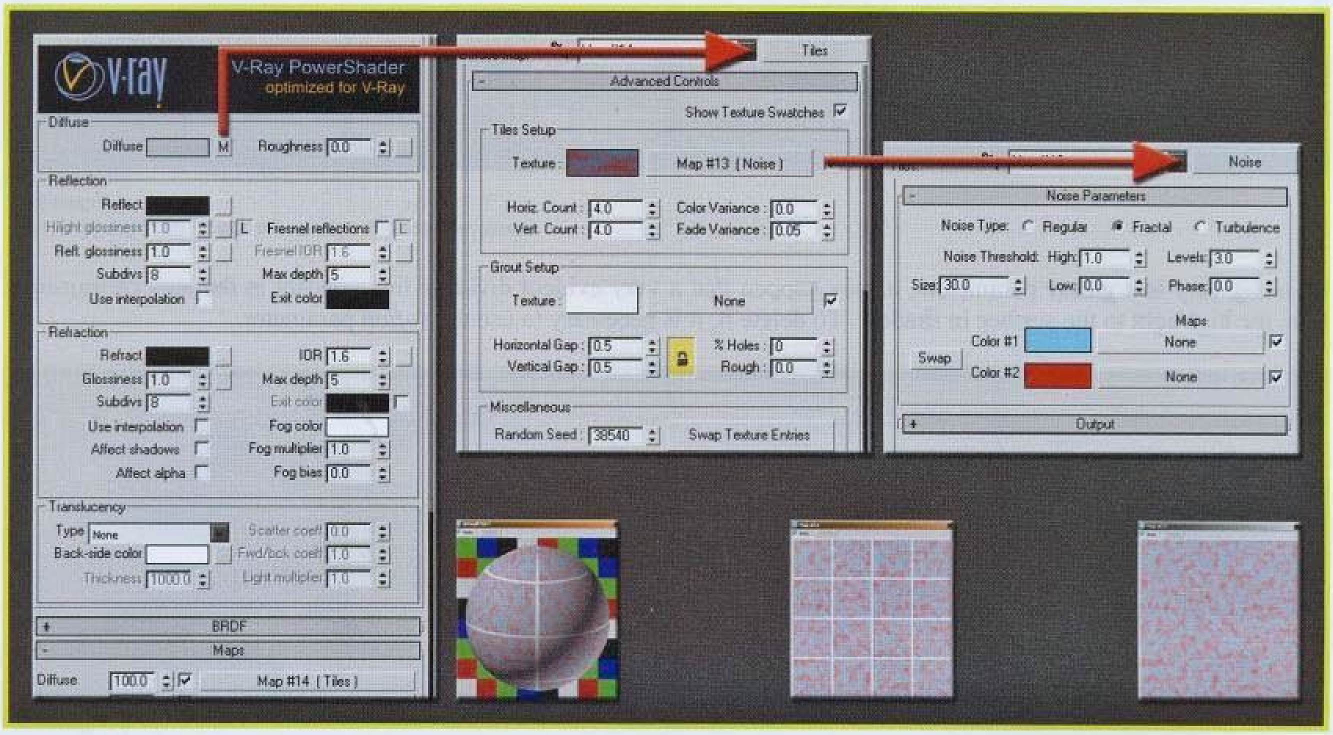
VRaySimbiontMtl

Chaosgroup have added a new shader, called ***VRaySimbiontMtl***. At the first sight it might seem a strange shader. But it has big potential. Before explaining it, it is necessary to describe a product which does not actually concern ***VRay***. Its name is DarkTree.

DarkTree, produced by the Darkling Simulations LLC, is a standalone program. In other words, it is a real software, like Photoshop or Autocad. In fact, once downloaded, one installs it and starts it. It is not a plug-in either of 3ds Max nor of ***VRay***: it is a program working autonomously.

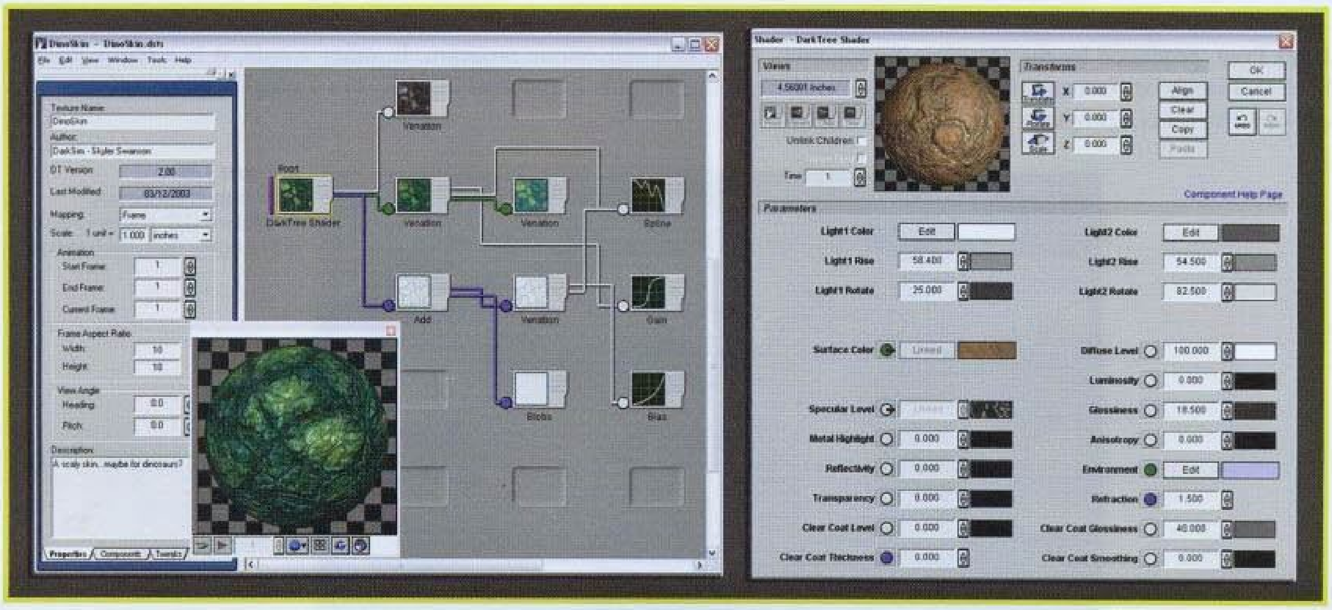
■ **Figure 9.99**
Homepage of the Darkling Simulations LLC, software house producing the DarkTree program.

It is a very advanced program, allowing one to generate very complicated and high-quality procedural shaders. To better understand this idea picture the Tiles map in 3ds Max. It is a procedural map. The produced texture is not a simple image taken by a scanner or from a digital camera, but it is generated via mathematic formulas, in a procedural way. Imagine that the color of the single tiles of the Tiles map is generated by another procedural map Noise. The result is a shader completely generated in a procedural way within 3ds Max.



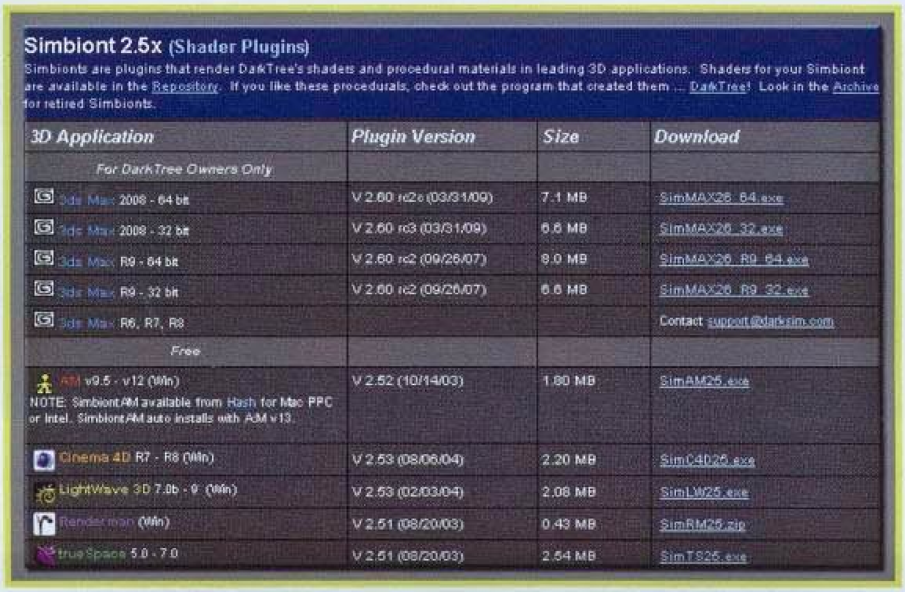
■ **Figure 9.100**
Accomplishment of a simple procedural shader with 3ds Max.

It is known that high-quality procedural shaders can be produced with 3ds Max, but, albeit its potential, it cannot equal the features that make DarkTree unique.
The software house Darkling Simulations is a program which is able to generate shaders with a photorealistic quality never seen until now thanks to some tree-diagrams called knots. By combining the 100 procedural maps available inside its "texture tree", it is possible to modify very high-precision textures, surfaces and shaders in real time.



■ **Figure 9.101**
Some screenshots of the DarkTree program.

The textures generated by DarkTree are computed in a 3D space, instead of coming from simple two-dimensional maps. Thanks to this feature, the DarkTree shader can be rendered at any dimension and resolution. In any case the maps can be exported in a bitmap.
Now one might ask: "How can these shaders be used?" A first answer has been already given in the previous lines. One way is to export a two-dimensional bitmap from the software. But in this manner the important procedural information would be lost. Therefore Darkling Simulations have created a tool for interfacing DarkTree not just with 3ds Max, but with all the most important 3D-graphics programs of the world. This tool is named Simbiont.



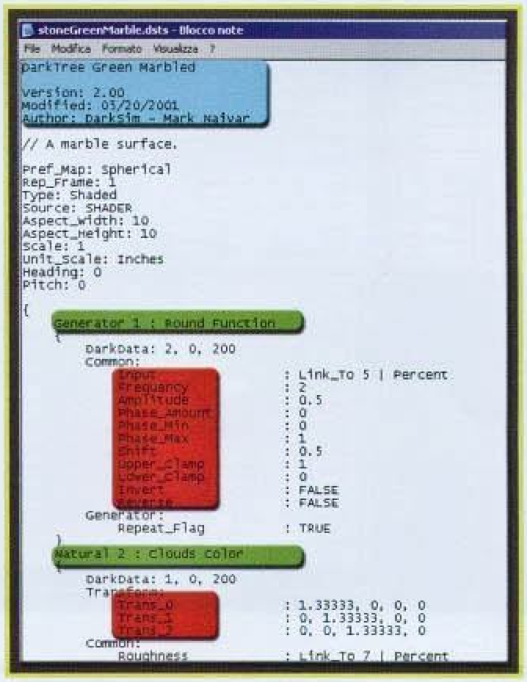
Simbiont 2.5x (Shader Plugins)

Simbionts are plugins that render DarkTree's shaders and procedural materials in leading 3D applications. Shaders for your Simbiont are available in the Repository. If you like these procedurals, check out the program that created them ... DarkTree! Look in the Archive for retired Simbionts.

3D Application	Plugin Version	Size	Download
For DarkTree Owners Only			
3ds Max 2008 - 64 bit	V 2.60 rc2c (03/31/09)	7.1 MB	SimMAX26_64.exe
3ds Max 2008 - 32 bit	V 2.60 rc3 (03/31/09)	6.6 MB	SimMAX26_32.exe
3ds Max R9 - 64 bit	V 2.60 rc2 (09/26/07)	8.0 MB	SimMAX26_R9_64.exe
3ds Max R9 - 32 bit	V 2.60 rc2 (09/26/07)	6.6 MB	SimMAX26_R9_32.exe
3ds Max R6, R7, R8			Contact support@darksim.com
Free			
AM v9.5 - v12 (Win) NOTE: SimbiontAM available from Hash for Mac PPC or Intel. SimbiontAM auto installs with AM v13.	V 2.52 (10/14/03)	1.80 MB	SimAM25.exe
Cinema 4D R7 - R8 (Win)	V 2.53 (08/06/04)	2.20 MB	SimC4D25.exe
LightWave 3D 7.0b - 9 (Win)	V 2.53 (02/03/04)	2.08 MB	SimLW25.exe
Renderman (Win)	V 2.51 (08/20/03)	0.43 MB	SimRM25.zip
trueSpace 5.0 - 7.0	V 2.51 (08/20/03)	2.54 MB	SimTS25.exe

■ **Figure 9.102**
The HTML page from which one can download Simbiont.

Simbiont is a plug-in allowing one to "read" the shaders generated and exported by DarkTree in programs such as 3ds Max, Cinea4D, LightWave etc... . DarkTree does not only export in .jpg o .bmp format, but also in .dsts; it simply is a proprietary format editable with Windows notepad, containing values which can be translated in procedural shaders thanks to Simbiont.

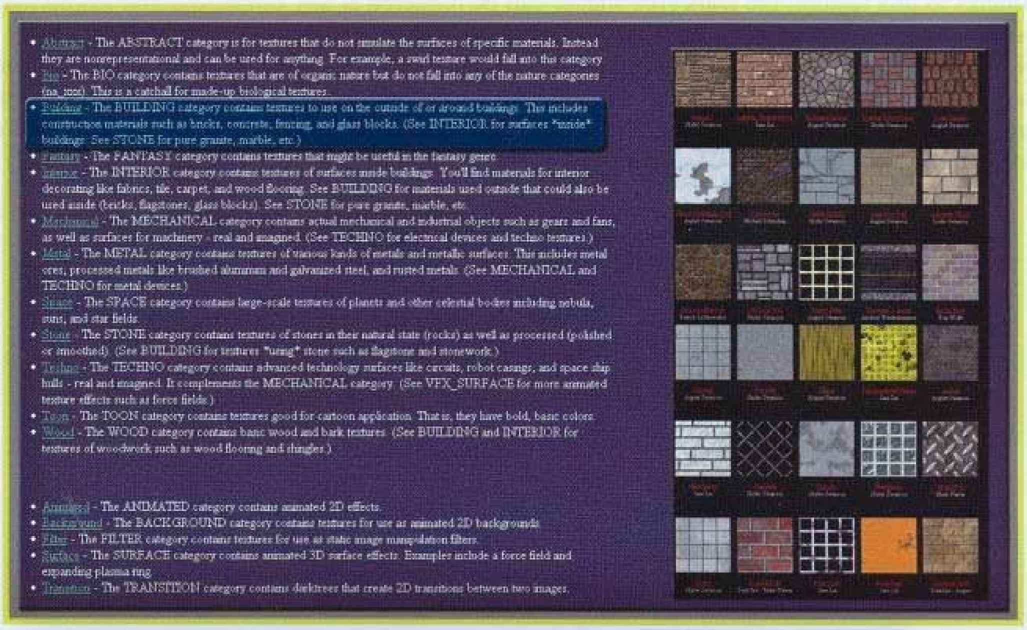


■ **Figure 9.103**
A .dsts shader opened with the notepad of Windows. The general data of the shader is highlighted in cyan, the knots in green, and the parameters of the knot in red. The sum of all these elements creates the final shader. Since it is a simple text file, its size is just a few KB.

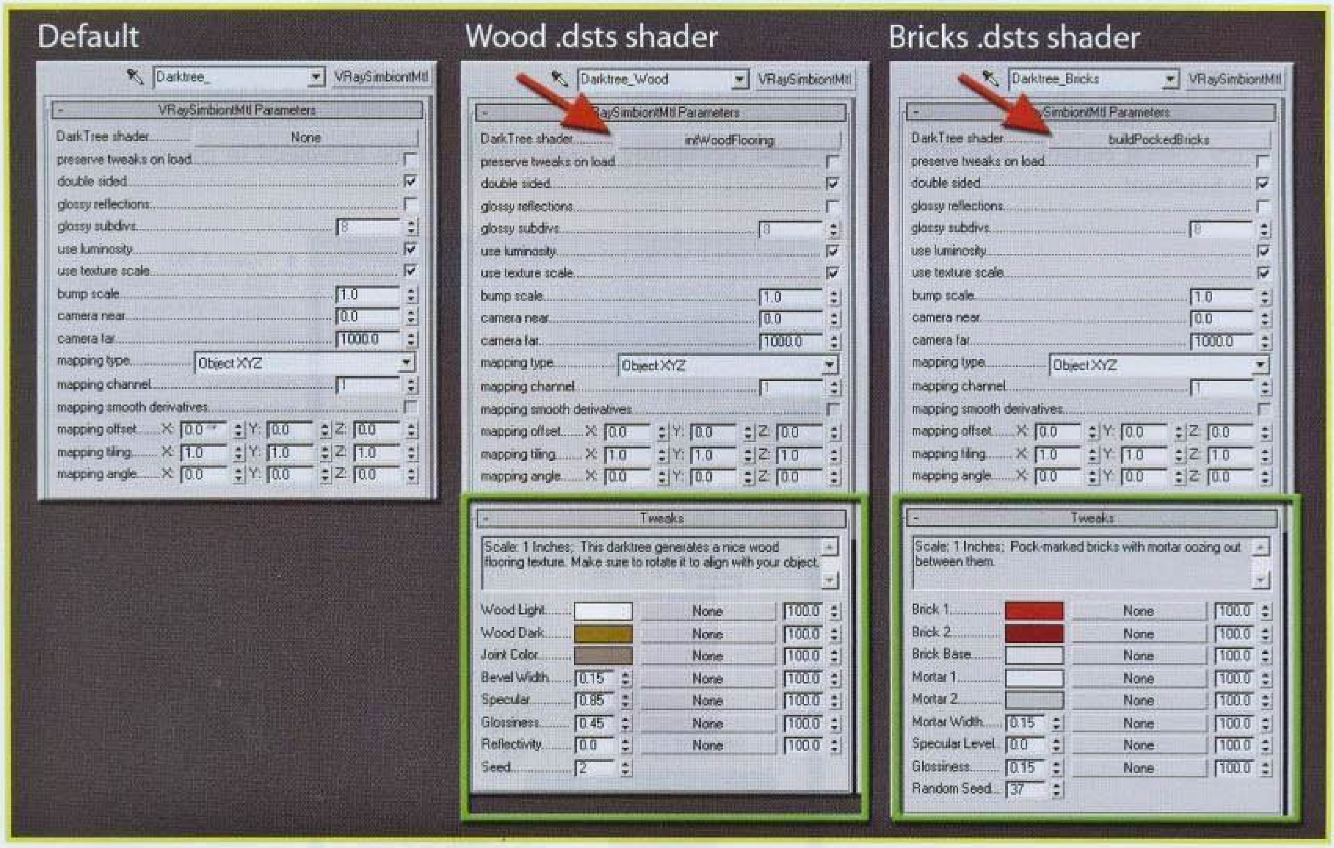
Why has it been necessary to create a ***VRaySimbiontMtl*** if Simbiont was already present? The same question could be asked with the Standard materials of 3ds max and the ***VRayMtl.*** But the latter is proprietary of ***VRay*** and therefore a higher compatibility, quality and speed ensue. The introduction of the ***VRaySimbiontMtl*** is mainly aimed at creating a complete integration between the shaders of ***DarkTree*** and ***VRay*** itself.

Obviously, to create a .dsts shader, one must purchase DarkTree, currently costing $419. But in the DOWNLOAD section of Darkling's site it is possible to download dozens of shaders, ready for being used with the ***VRaySimbiontMtl***.

■ **Figure 9.104**
Some of the free shaders .dsts available. In the image it is possible to see the preview of the Buildings shaders.



The ***VRaySimbiontMtl*** appears to be quite a complex shader, but it becomes even more complex when a .dsts file is loaded within it.



■ **Figure 9.105**
Starting form the left, the ***VRaySimbiontMtl*** without any .dsts file loaded. In the middle the Wood shader, and on the right the Bricks shader.

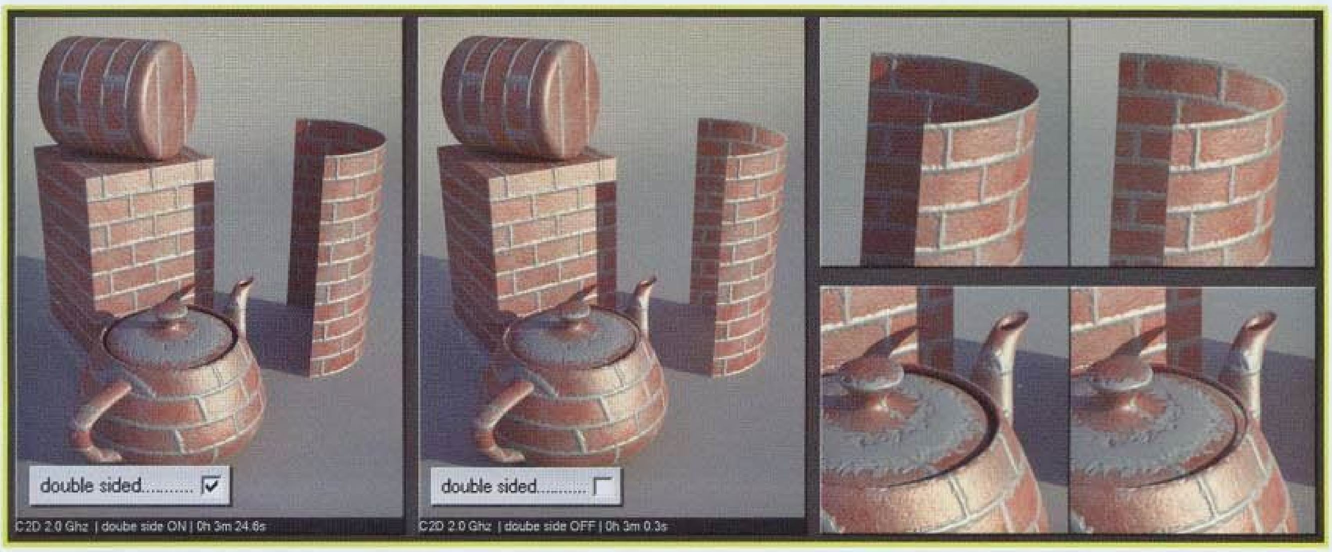
The ***VRaySimbiontMtl*** shader contains parameters which are quite intuitive to some degree, such as the ***glossy reflection*** or some ***mapping*** parameters. The most evident thing is the appearance of a new rollout after the loading of the .dsts files, called ***Tweaks***. The parameters of the ***Tweaks*** section are values "included" in the same .dsts shader and made available by whoever programmed the shader via the DarkTree software. For example, in the Bricks shader, the parameters for changing the color of the brick, the mortar and its thickness are exposed.

Like the Wood shader, parameters are different and correspond to the color of the veins, of the space in between the plankings, and the reflection of the axis. The extreme power of DarkTree, the generated .dsts shaders and the ***VRaySimbiontMtl*** is comprehensible. All of this with a considerable result.

DarkTree shader - in this slot the .dsts file, generated by the Darktree program, can be loaded.

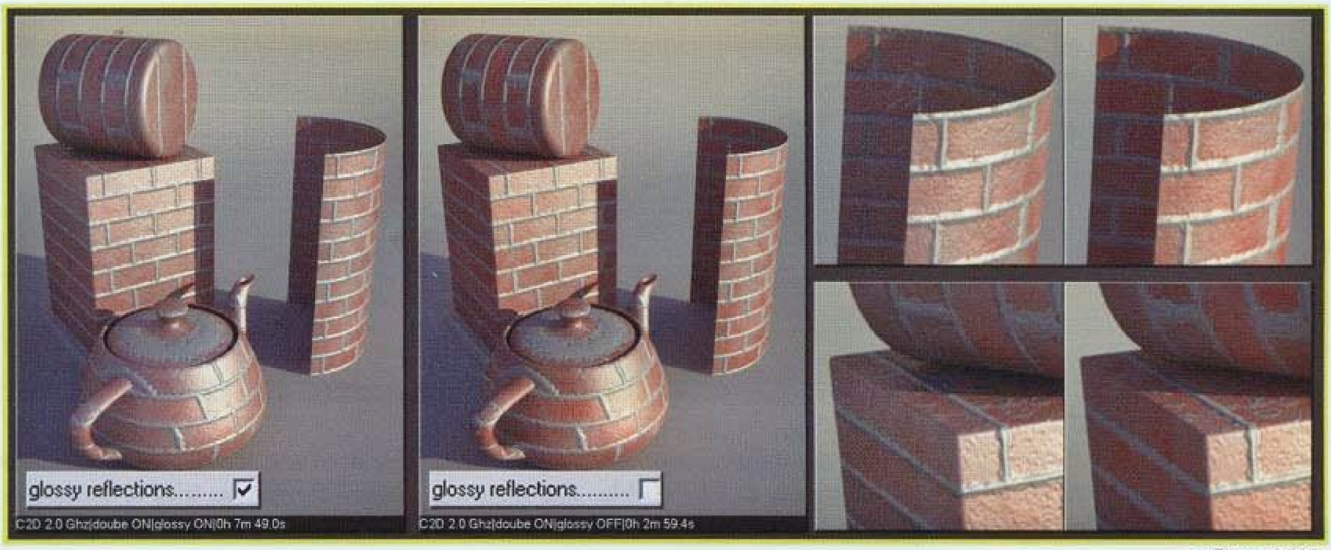
preserve tweaks on load - by activating this parameter when, for any reason, one wants to find the .dsts file in use, the parameters of the ***Tweaks*** rollout are not modified. Vice versa, the default parameters of the .dsts shader are restored.

double side - by activating this parameter for geometries without thickness, the selected shader is used on both the faces.



■ **Figure 9.106**
Notice how the deactivation of the ***double side*** parameters hinders the rendering of some geometries.

glossy reflections - in this specific example, the shader has a certain level of specularity, as one can notice on the teapot. By activating the ***glossy reflection*** parameter, ***VRay*** will compute the physically correct smoothed reflections beyond the "fake" highlights. In practice, it is what happens by activating or deactivating the ***Highlight glossiness*** in the ***VRayMtl***.



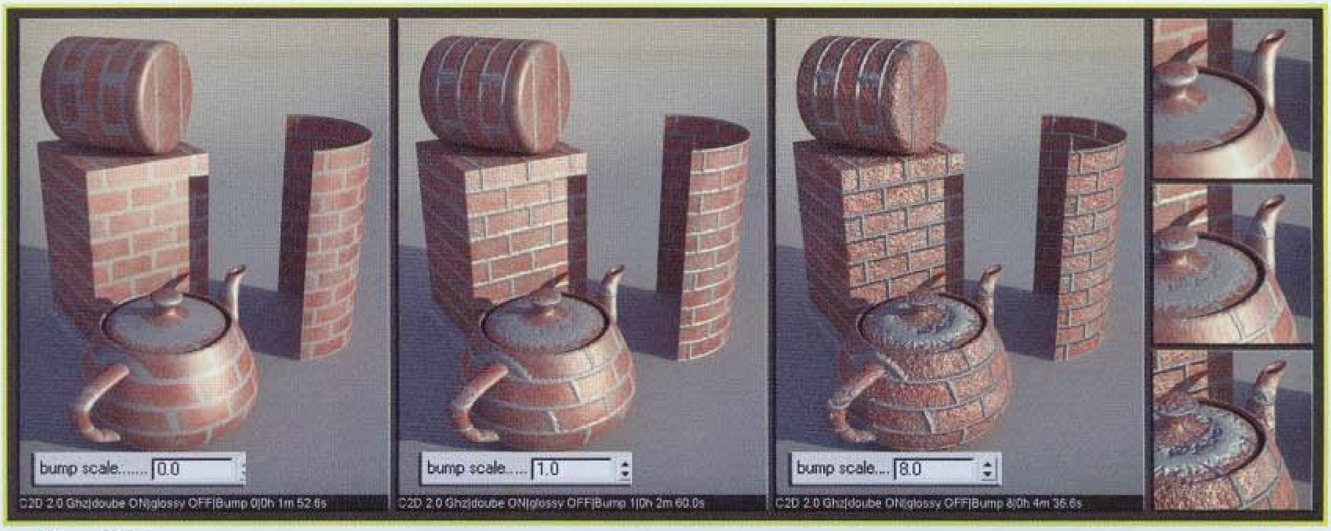
■ **Figure 9.107**
In this specific example, the ***VRaySky+VRaySun*** will reflect on the geometries. The reflection is very smooth, and the ***glossy*** is not so perceivable although it is present.

glossy subdivs- thanks to this option one may control the quality, that is, the presence or not of noise, in the smoothed reflections. This parameter corresponds to the ***Subdivs*** in the ***VRayMtl***.

use luminosity - it can happen that a shader produced by DarkTree contains the self-illumination channel. In this case the ***use luminosity*** parameter can be used, which lets one activate or deactivate this channel, where the glow, and therefore the self-illumination, could give problems in the computing of the GI.

use texture scale - many DarkTree shaders have properties defined in world units. For example, a texture of a floor might be 2m x 2m. this is supposed to be the equivalent of the option of 3ds Max named Real-world. By activating the ***use texture scale***, ***VRaySimbiontMtl*** uses the current unit system of 3ds max and scales the .dsts shader in such a way to adjust it to the current unit of measurement.

bump scale - many DarkTree shaders have the bump channel. If it is present, the channel increases or decreases in intensity by modifying the ***bump scale***.

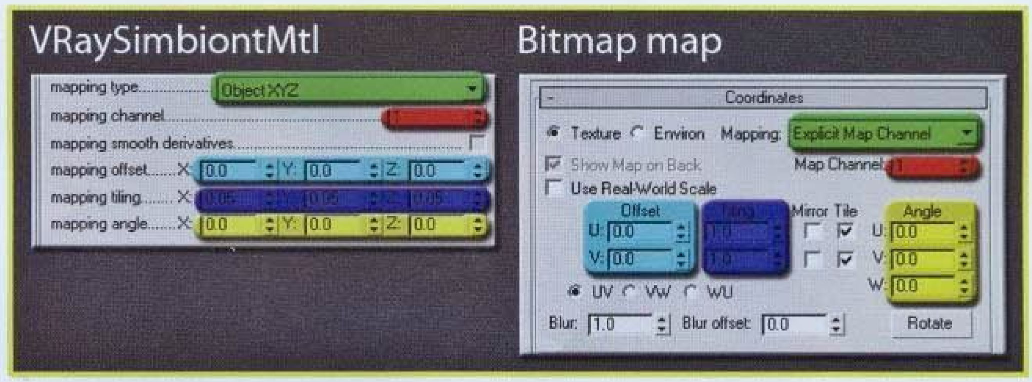


■ **Figure 9.108**
Increase in the intensity of the bump channel.

camera near/far - some DarkTree shaders may have a LOD (**L**evel-**O**f-**D**etail) system within it, allowing one to reduce the details of the textures as the shader gets further and further from the camera. This system has the purpose of speeding up rendering time. If the shader has this function active, the ***camera near/far*** parameters are used to manage the start and end point of the LOD system.

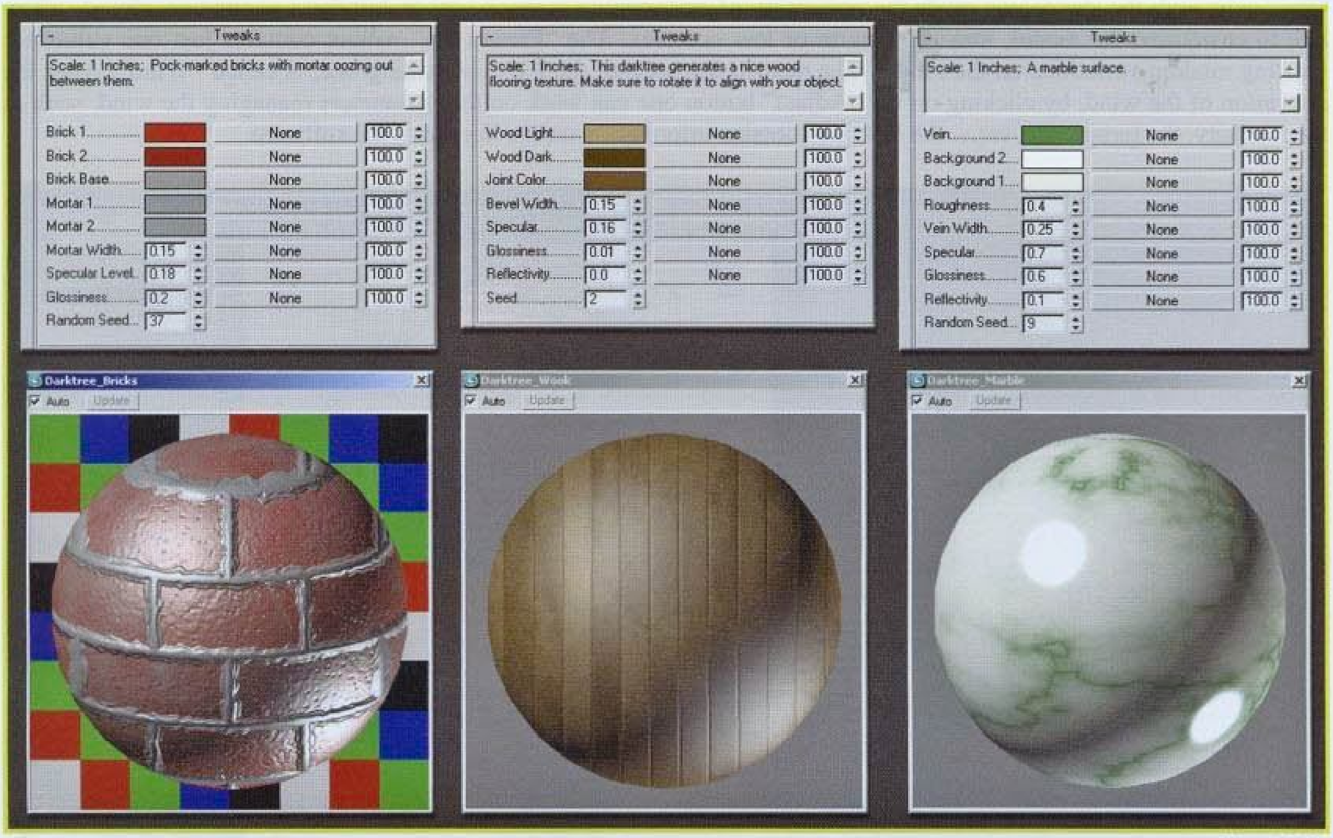
The next parameters, that is the ***mapping type***, ***mapping channel***, ***mapping offset***, ***mapping tiling***, ***mapping angle***, have the same functions as the parameters located in the maps of 3ds Max.

■ **Figure 9.109**
Parameters controlling the position, the repetition and the angulation of the DarkTree shader.



mapping smooth derivatives - when the Explicit map channel coordinates are used and the shader has a lot of bump, sometimes the facets of the basic geometry can be visible. The ***mapping smooth derivatives*** tries to avoid this kind of problem. On the other hand there is a higher usage of memory.

Tweaks - the parameters of the ***Tweaks*** rollout are variable from shader to shader and might even not be present in a DarkTree shader. It all depends on who has generated it. Beyond the presence of a brief descriptive text, most of the time they allow the management of the textures, their size, the intensity of the highlights and the reflection.



■ **Figure 9.110**
Three examples of shaders with the relative parameters.

VRayProxy

Chaosgroup has decided to widen the possibilities offered by its proxies with *SP2*. From now on it will be possible to insert in the scene not just static proxies, but animated ones as well! For example, it is possible to create a whole forest with blowing trees, or an animated crowd, with very few hardware resources. The procedure for the creation of animated proxies does not differ that much from static ***VRayProxy***.

1. First of all one has to model the object to be animated. It can be a deformed geometry, such as a blowing tree or a animated character.
2. Then one must select the mesh to be exported. In the case of the tree, one selects the complete geometry, and for the person obviously just the mesh, without bone structure or anything else.
3. The mesh is imported as ***VRayProxy*** and the values of the animation are set.

In the case considered, the plug-in Tree Storm v5.9 for 3ds max 2008 is to be used for creating the blowing tree. It is a plug-in by Onyx Computing (www.onyxtree.com), which allows one to manage and import directly into 3ds Max the geometries created with the standalone program OnyxTree, a parametric modeller for the generation of trees. The functioning of this software is very similar to the idea used by the Darkling Simulations with its DarkTree standalone and the Simbiont plug-in. The peculiarity of this modeller relies in the high quality of the models, in addition to the possibility of animating them realistically with the wind, itself included in the proprietary file .tr5 as well.

This is not a chapter dedicated to Onyxtree, but it is necessary to know how it works to understand how to generate an animated ***VRayProxy***. The plug-in for 3ds Max contains 6 rollouts. The first one, "Choose", allows one to choose its file .tr5. The rollout "Preview Mode" offers the possibility of choosing how to visualize the model in the viewport of 3ds max. One may choose between the simple, medium and complete mode. As one can see in the next image, an Onyx model is very heavy. Usually it is at least composed of 10,000 polygons and it may get over 400,000. It is therefore comprehensible how difficult it is to move and manage even just one tree in the viewport of 3ds Max in complete mode: imagine a forest. That is why it is possible to choose different visualizations. With the "Polygons" rollout one can work on the number of polygons composing the model.

Obviously, with the standalone mode the possibility of customizing is much higher, but these parameters already allow one to customize the model, making it more or less complex. The "Parameters" rollout contains just one value, for creating random variations on the model. The next-to-last rollout, the most interesting one, is called "Wind". After the activation of the wind, by clicking on the "Adjust" button one can work on the parameters managing the wind, such as the intensity, the turbulence, etc… the last one contains information on the license of the software.

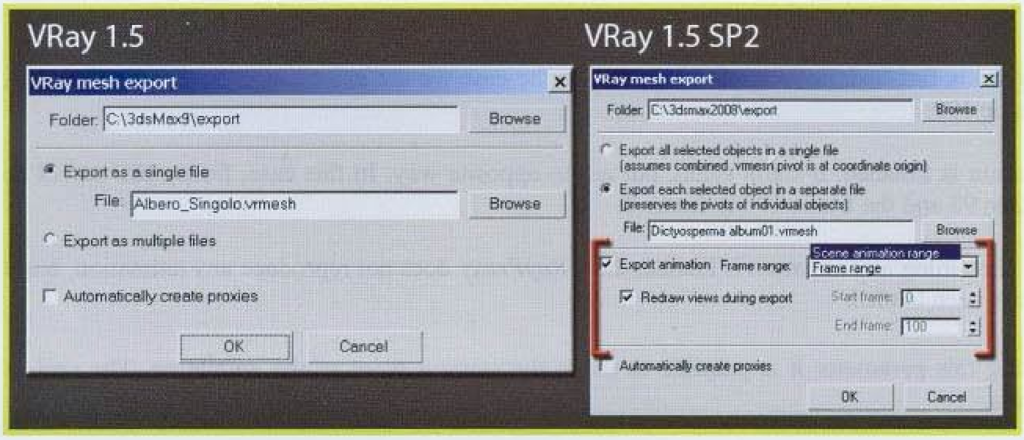
■ **Figure 9.111**
The interface of the Tree Storm plug-in. Notice how just one tree contains more than 35,000 polygons.

Here below, the model with the wind activated.

■ **Figure 9.112**
Activation of the wind blowing from right to left.

The total length of the animation is 100 frames, during which the wind bends branches and leaves. The rendering does not give many problems when just one model is present, but problems of memory usage may arise when the geometry is multiplied many times. At this point, the animation is to be "captured" in a *.vrmesh* file, inside which all the data relative to the animated geometries are contained: in this case trunk, branches and moving leaves.

With the right button of the mouse, in the Quad Menu, the ***V-Ray mesh export*** is selected. The tool for the creation of the ***VRayProxy*** appears. Compared to the previous version, some new tools can be noticed, aimed at the creation of the animated ***VRayProxy***.



■ **Figure 9.113**
The new section devoted to the generation of animated ***VRayProxy*** is highlighted.

Export animation - by activating this option, ***VRay*** exports the selected animation. It is possible to choose to export the whole animation, identified in the timeline, or to choose a certain range which can be defined with to the parameters ***Start frame*** and ***End frame***.

Redraw views during export - if this option is active, it is possible to observe the selected animation in the viewport while the ***.vrmesh*** is being created. It is useful if the animation to be exported is very heavy and the refresh of the animated geometry requires a lot of resources. In this case, deactivation makes the export process faster.

At this point one has a ***.vrmesh*** file available, containing all the data of the animated palm. A little note. The .max original file, that is the one actually generated by the Tree Storm plug-in, weighs 500 KB. The version converted to Poly weighs 2.9 MB, while the ***.vrmesh*** file containing just one tree roughly 1.2 MB. The ***.vrmesh*** file of entire animation roughly 150 MB.

■ **Figure 9.114**
Size of the employed files.

The reason the animation is not 120 MB in size (1.2 MB x 100 frames) is that inside the animation some other information, such as the ***VRayVelocity*** channel, is stored.

At this point the animation file is imported by using the ***VRayProxy*** tool. Pleasant surprises can be found here as well.

■ **Figure 9.115**
New parameters in the ***VRayProxy*** introduced with ***VRay 1.5 SP2***.

point - by selecting this option, the geometry (in this case a tree) is represented by a point located at one's pivot. This mode is useful if there are too many ***VRayProxies*** which prevent a correct visualization of the scene in the work viewport.

Playback - it is possible to modify the way the animation is reproduced for the ***VRayProxy***.

Loop - once finished, the animation starts again. In this case, its playing time is 100 frames. If one were to use the tree in an animation 1,000 frames long with the ***Loop*** method, the frame 101 would be equal to the frame 1, the 102 to the 2, and so on.

Play once - once the animation is finished, nothing else happens. In the case we are examining, the frames 101, 102, 103.... would be equal to frame 100.

Ping-pong - once the animation is finished, it starts again, but in the opposite way. In this case, frame 101 would be equal to frame 99, frame 102 to 98 and the frame 199 to frame 1.

Still - even if a ***.vrmesh*** file containing an animation is loaded, the ***VRayProxy*** does not reproduce any animation, but it behaves as if a static proxy has been loaded.

Offset (frames) - by using this parameter it is possible to move the starting point of the animation. The unit of measurement used is the frame.

Speed - this parameter allows one to modify the reproduction speed of the animation.

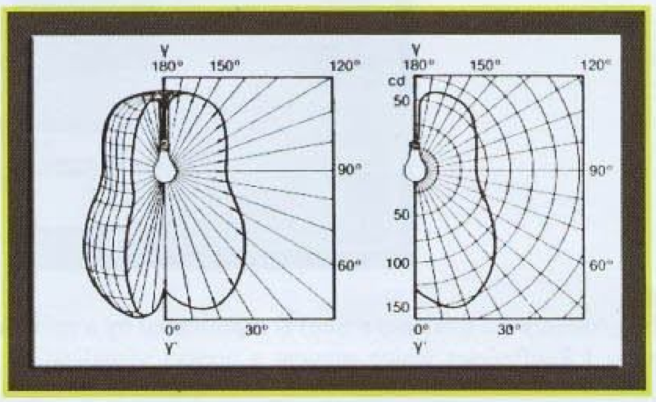
One last note about the animation and the reproduction. Usually the real time computing of animated objects is very slow. By using the ***VRayProxy***, instead, since the information has been already computed previously and since the object represented in viewport by the ***VRayProxy*** contains a small number of polygons, it is possible to reproduce in real time and fluently any animation. In this case, even having a DualXeon QuadCore and 8 GB Ram, the reproduction of the animation with the Tree Storm plug-in was roughly 5 FPS. With ***VRayProxy*** the animation is in real time, that is 30 FPS.

Now you have all the information for creating an entire animated forest!

VRayIES

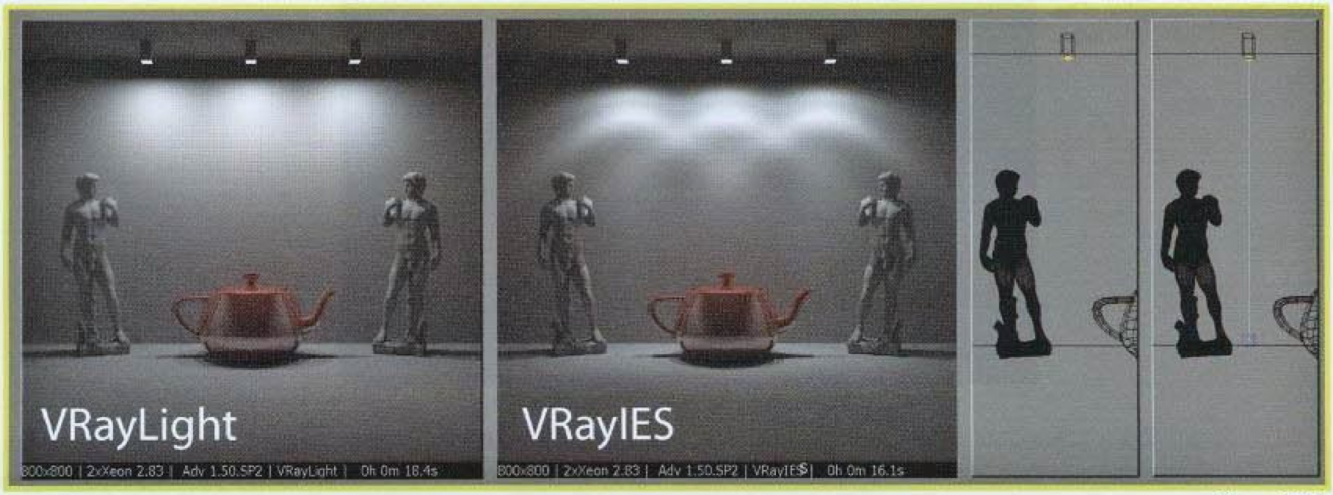
The IES files are generally produced in companies which manufacture light appliances, and contain data of photometric curves of a light source. In fact, the photometric curve is a tool allowing one to graphically represent how a light source emits light in the space.

A photometric curve can be associated to any object emitting light, from a simple bulb to a large lighting device. The photometric curve of a lighting device allows one to analyze, and in Computer Graphics to simulate as well, its impact on the surrounding environment. It is necessary to measure the luminous intensity of the device to obtain its photometric curve, or, in other words, it is necessary to measure with how much intensity the source emits light in one specific direction. Essentially, it measures what would happen if one moves around the source with an instrument measuring the intensity of the emitted light from different angulations. In such a way a three-dimensional graph would be obtained, which, thanks to the ***VRayIES***, can be translated into information used to simulate exactly the luminous device associated to the .ies file (for a deep analysis of this topic, see CHAP. 6).



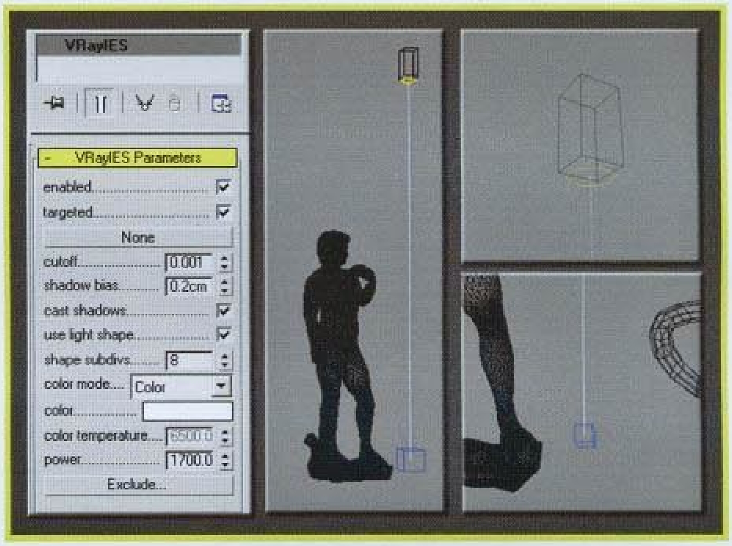
■ **Figure 9.116**
Three-dimensional and two-dimensional graphs of a photometric curve.

Why should one use the *VRayIES*, since many light sources are already available? The reason is that these lights are more realistic. Observe the following image.



■ **Figure 9.117**
Comparison between a ***VRayLight Plane*** and a ***VRayIES***.

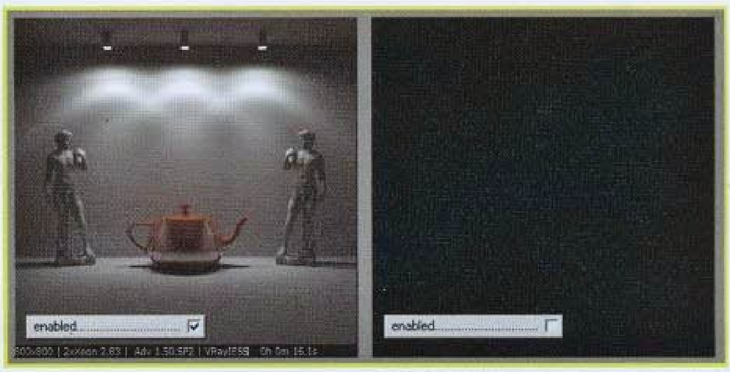
The scene, characterized by a strong bright/dark contrast, makes the difference between a normal *VRayLight* light and a *VRayIES* even more evident. In this last case the light source produces a shape projected on the wall. It is as if a lamp were inserted inside the luminous body, which creates very detailed effects. In truth nothing is actually present on the scene. These particular play of lights are "included" in the .ies file and exactly reproduced by *VRay*.



■ **Figure 9.118**
Interface of the ***VRayIES*** and graphic representation in the working area.

The management panel of *VRayIES* contains 13 parameters, of which most are already included in the normal *VRayLights*. In the viewport it appears as a *VRayLight* with its target.

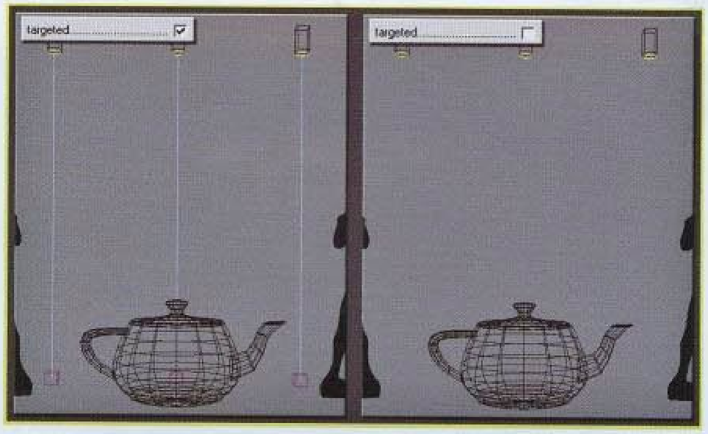
enabled - thanks to this parameter it is possible to switch the *VRayIES* on and off.



■ **Figure 9.119**
Activation and deactivation of the light source.

targeted - by deactivating this option, the ***VRayIES*** no longer has a target and is going to behave as a Free Light.

■ **Figure 9.120**
Activation and deactivation of the target.



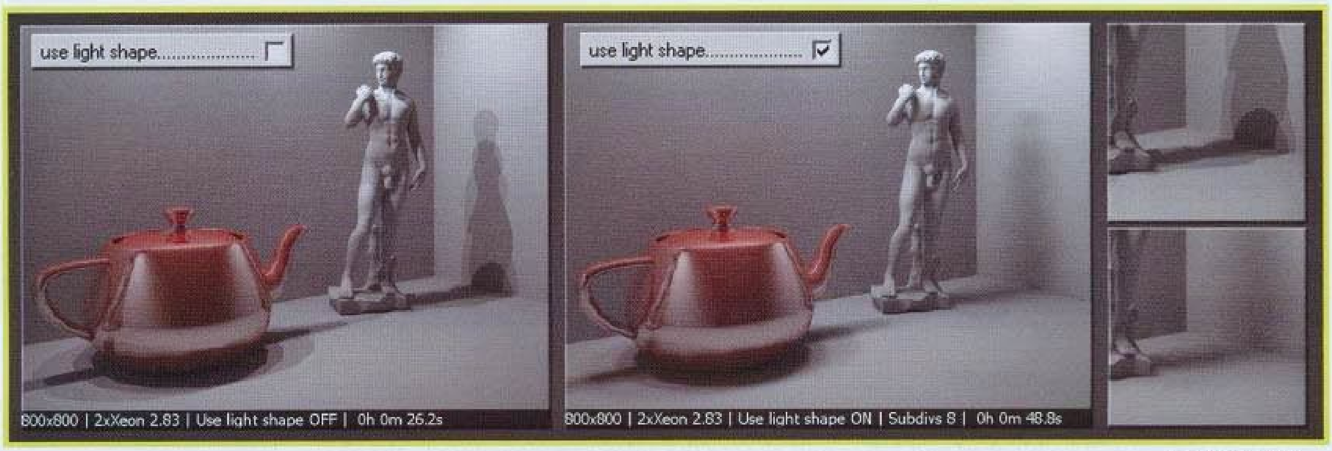
None - in this slot it is possible to load the .ies file. Notice how the ***VRayIES*** icon adapts to the new photometry as the .ies file changes. Beyond this, also the luminous intensity, expressed in lm and represented by the power parameter, is updated to the new file. In the next example three .ies files are going to be used and the photometric light of 3ds Max will join the ***VRayIES***.

■ **Figure 9.121**
Example of employment of three photometries.

By analyzing the previous image it is possible to see how the first two .ies files are photometric files with symmetric curves, whereas the last one is asymmetric. Unfortunately the *VRayIES* do not show the photometric graph in the viewport as lights in 3ds Max do. For the symmetric photometric files this is not a big problem, but it is for the asymmetric ones, for which the correct position must be found only by trial and error. Furthermore, the visualization of the curves in the working windows allows one to have an idea of how the light is going to behave. This phenomenon is clearest in the second example, where the light source, very thin, long and slender, also produces a cone of concentrated light in the render.

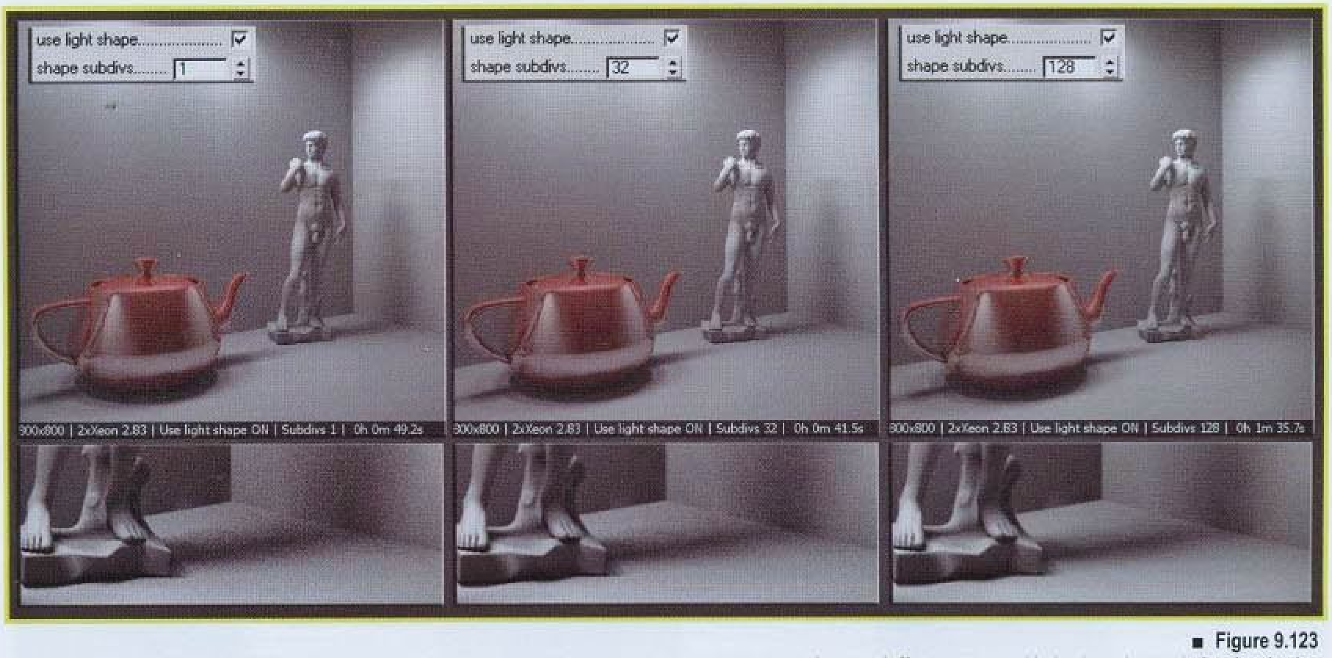
cutoff - **shadows bias** - **cast shadows** - these three parameters are the same as the ones located in *VRayLights*. For an in-depth explanation, see CHAP.6.

use light shape - inside them the photometries have information concerning the size of the light source. The *VRayIES* are able to read this information and can use them to generate *Area Shadows*. By activating the option *use light shape*, the *VRayIES* allow the forming of shadows with smooth sides. This requires a higher rendering time.



■ **Figure 9.122**
Generation of the ***Area Shadows*** by using the ***VRayIES***. Notice how the rendering time is nearly doubled.

shape subdivs - parameter controlling the quality of the *Area Shadows* generated by the *VRayIES*.

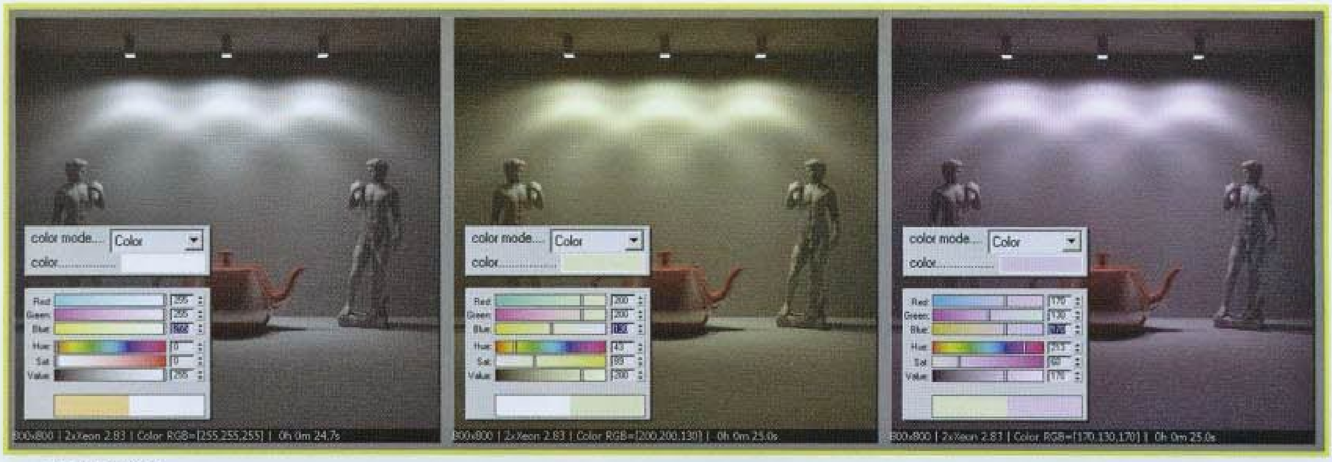


■ **Figure 9.123**
Increase of the quality and reduction of noise obtained by means of the increasing the ***shape subdivs*** parameter. Notice how the rendering time is also increased.

color mode - with this pull-down menu it is possible to choose the mode of coloring the light generated by the *VRayIES*. The coloration may be obtained by choosing directly the RGB color from the classic Color Selector of 3ds Max, or by inserting the temperature of the color. In this case, by setting the value in Kelvin degrees, *VRay* is going to set automatically the RGB color. The latter method leads to a more realistic result.

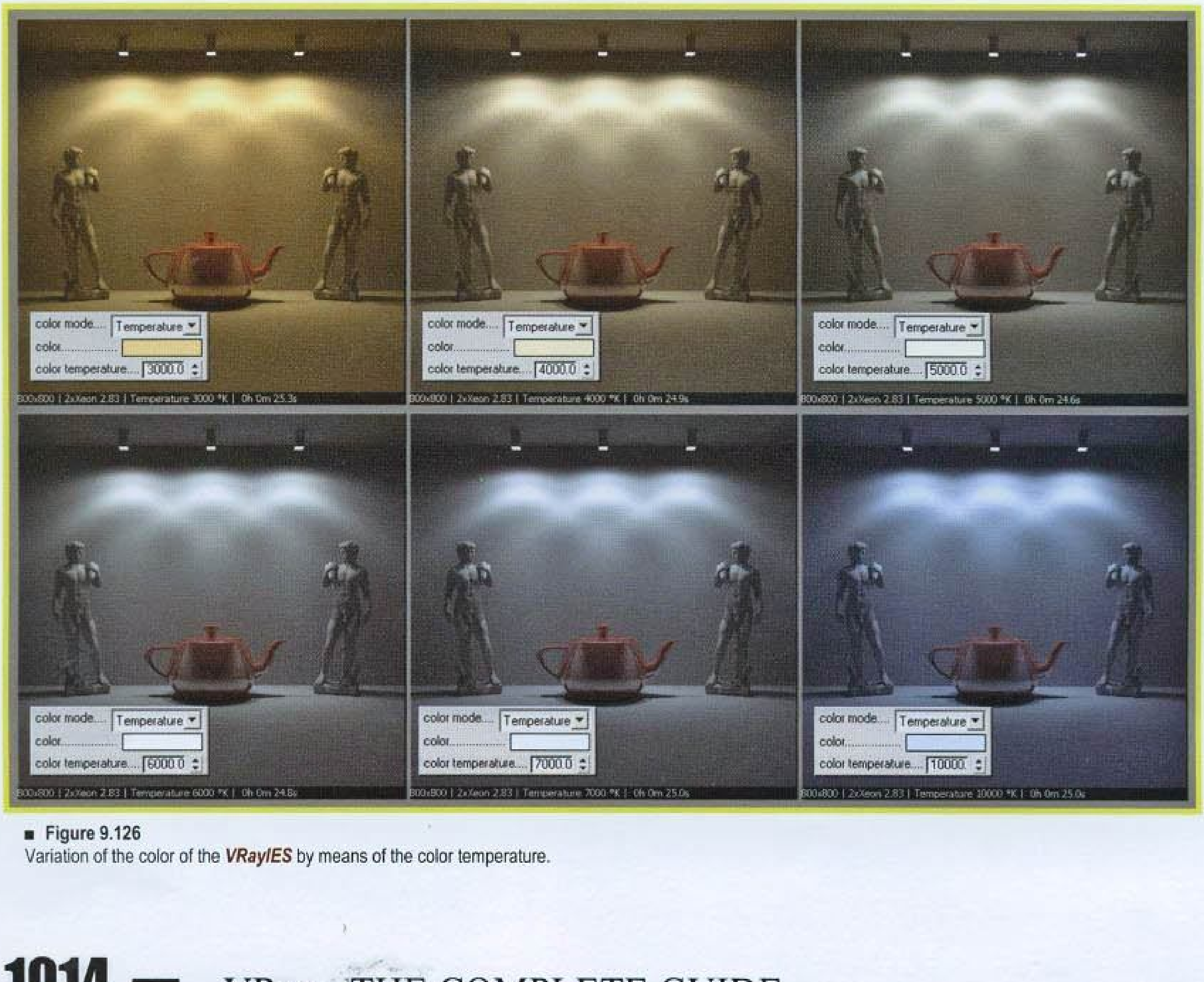
■ **Figure 9.124**
The pull-down menu for the choice of the coloration mode of the VRayIES.

color - if ***color mode*** is set to ***Color***, the user can choose manually the RGB value of the ***VRayIES***.



■ **Figure 9.125**
Manual variation of the color of the ***VRayIES***.

color temperature - if ***color mode*** is set to ***Temperature***, the user has to insert the temperature of the light source. The color selector will be blocked and *VRay* will choose the color corresponding to the chosen temperature.



■ **Figure 9.126**
Variation of the color of the ***VRayIES*** by means of the color temperature.

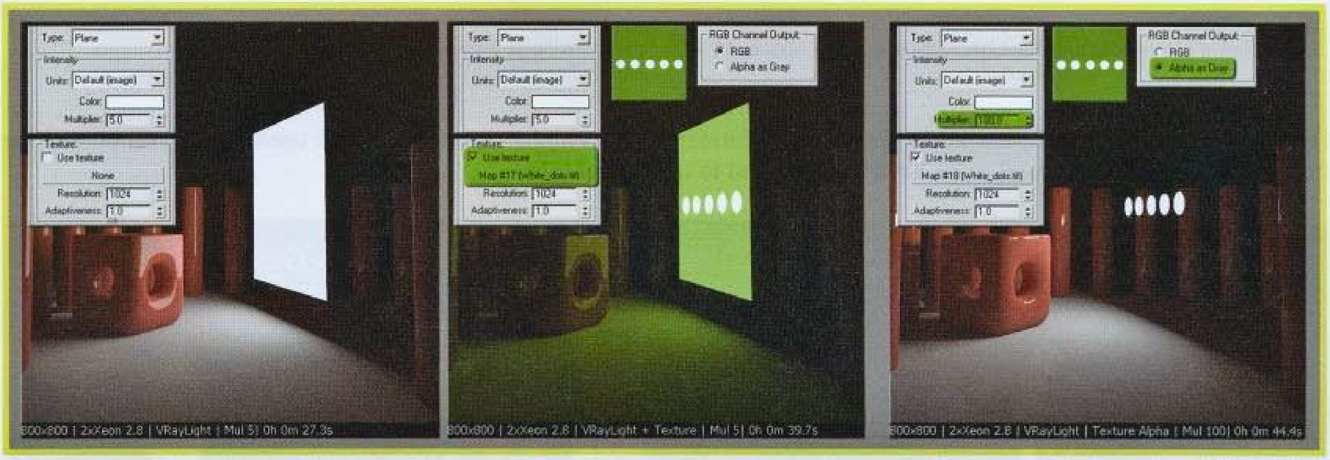
power - once the .ies file is loaded, *VRay* recognizes the power of the light source, and inserts this value in the *power* parameter. In any case, the user can always intervene by manually increasing or decreasing the intensity of the *VRayIES* light.

■ **Figure 9.127**
The parameter ***power*** corresponds to the classic ***Multiplier*** for ***VRayLights***.

Exclude... - this parameter is the same as the one in the *VRayLights*. For a thorough explanation see CHAP. 6.

VRayLight

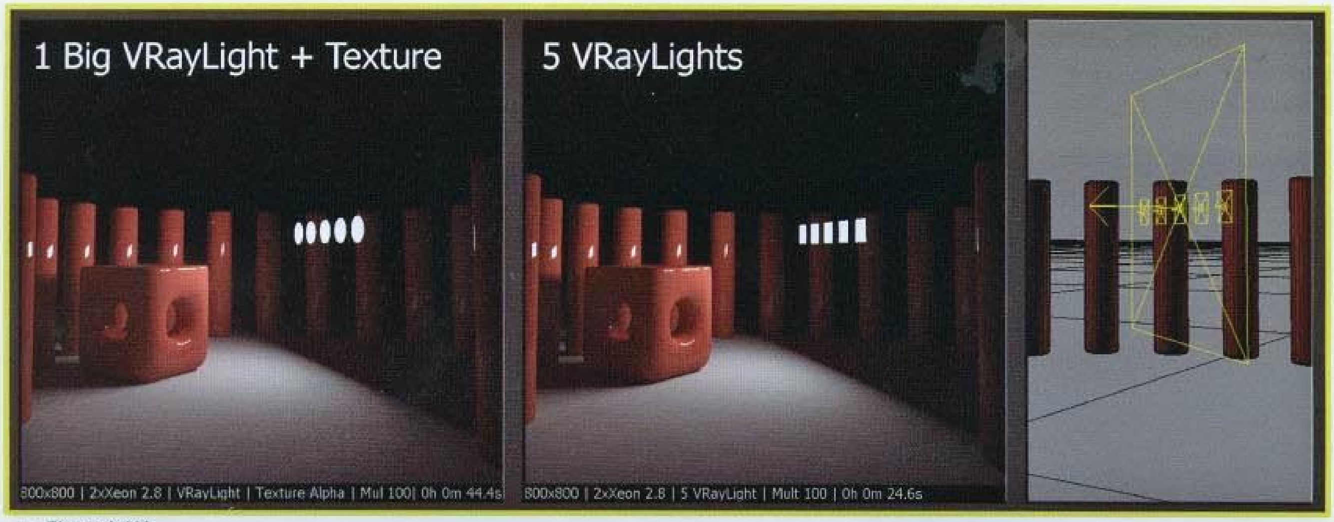
The ***Plane*** type ***VRayLight*** is now able to use textures, procedural or not, as light generators. The idea is very similar to the ***Dome light***, by means of which a map can emit luminous intensity.



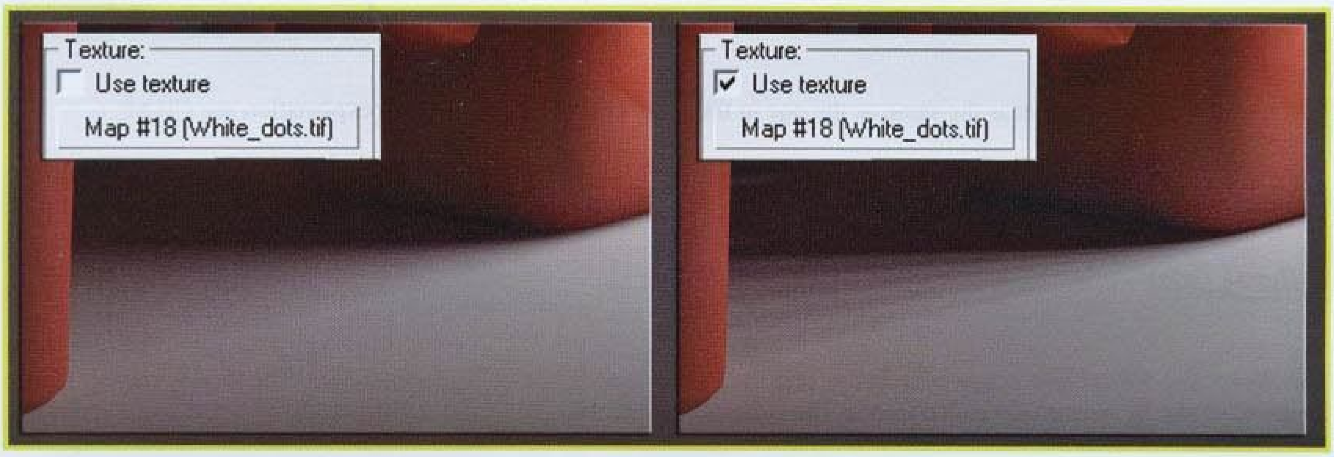
■ **Figure 9.128**
Employment of textures for the generation of lights. The activation of the alpha channel takes place in the Material Editor, within the Bitmap map.

In the first image there is nothing new. A ***VRayLight Plane*** illuminates the scene. In the second image, instead, a texture in .tiff format has been used, prepared with Photoshop. This map contains the alpha channel, a channel that *VRay* recognizes, and thanks to which it has been possible to "cut", in order to create five spotlights.

To compare the result, the same scene has been rendered with five small *VRayLights*. The result is the same, although it contains more noise due to the bigger size of the *VRayLight*. Notice how the point sources, created by means of a texture, actually create realistic shadows, as if they were really five different sources.



■ **Figure 9.129**
Comparison of two different kinds of light source.



■ **Figure 9.130**
Area shadow generated by the ***VRayLight Plane*** by using the texture with white dots. Notice the generation of very detailed soft shadows.

Here below are other examples. In the first image a bitmap with its alpha channel has been used. In the second and the third ones, instead, procedural maps have been applied to the ***VRayLight***.

■ **Figure 9.131**
Example of use of the new texturization function.

VRayShadowMaps

The last change in the lights section is the ***VRayShadowMap***. This type of shadow has been introduced in order to reduce the flickering problem as much as possible in the presence of ***VRayFur***. Until now, by using the normal ***VRayShadow***, it could happen that artefacts formed during the animations. Thanks to these shadows this problem has been solved. The ***VRayShadowMap*** work with all the geometries of 3ds Max, ***displacement*** and ***VRayProxy***.

■ **Figure 9.132**
A new type of shadow, the ***VRayShadowMap***, is now to be found in the pull-down menu Shadows, located in the General Parameters rollout of the default lights of 3ds Max. the technology used in this particular shadow is very similar to the one used for the normal Shadow Map of 3ds Max. The main difference is that these shadows have been written to operate with all the tools of VRay. Therefore, once selected, a new rollout appears, called ***VRayShadowMap*** parameters, by means of which one can modify the shadows.

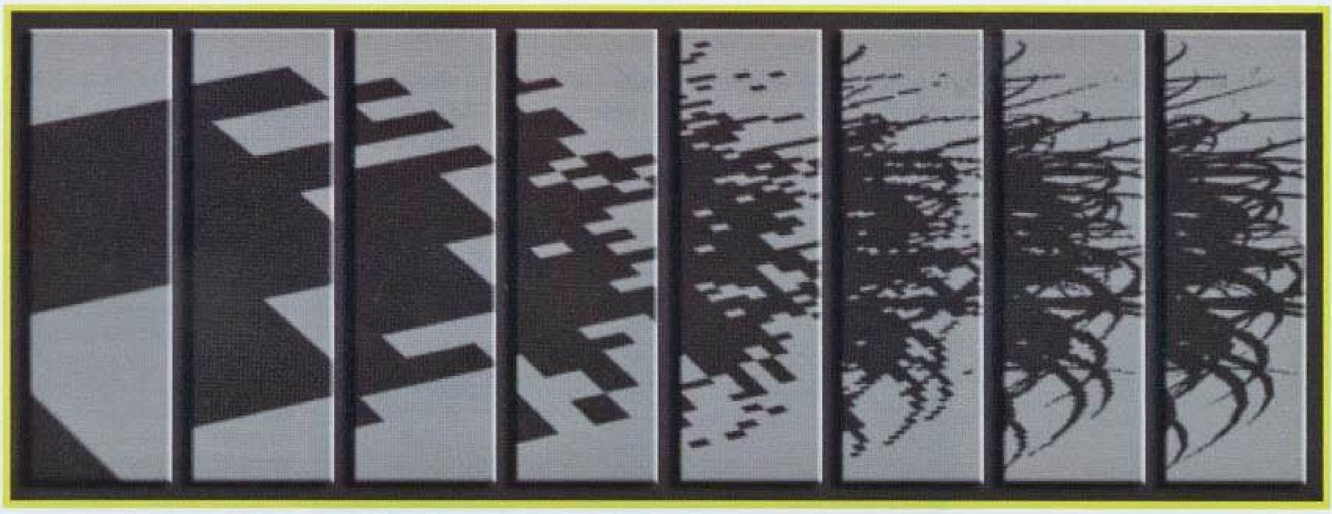
mode - this pull-down menu allows one to choose among two computing modes: ***Single frame*** and ***Fly-through***.

Single frame - with this method the map for the generation of shadows is computed for each new frame, even if no object moves on the scene or no variation of the shadows is present.

Fly-through - in this case the shadow map computed in a certain frame, for example the first one, is to be used for the second one as well, and in case of changes, displacements etc.. information is to be added or removed according to the need. This mode is useful with fly-through animations and allows one to save computing time.

resolution - this parameter sets the size of the computed shadows map. The size determines the amount of subdivisions of the map. The higher the value is, the more detailed the map is.

■ **Figure 9.133**
Shadow maps generated with different resolutions. Notice how rendering time increases as the resolution increases.



■ **Figure 9.134**
Detailed image.

bias - this parameter allows one to move the map closer or further from the object generating it. For details, see CHAP.6.

filter - parameter determining the size of the area to be pondered within the shadow. This makes it possible to mask problems of low-resolution maps, making the shadow softer. But in this case one obtains a loss of detail in the shadow as well.

■ **Figure 9.135**
Notice that the shade of the shadow increases by increasing the ***filter*** parameter. As a consequence, there is a loss of detail of the shadow, and an increase in the rendering time as well. In this case a 512 resolution map has been used.

filter method - this pull-down menu allows one to choose two computing methods for the generation of the shadow map.

Exact - it is the slowest one, but it produces also the least noise, which is due to the filter parameter, during the interpolation of shadows.

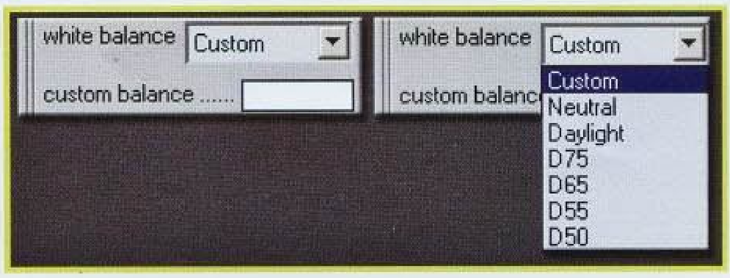
Fast - within some limits, it is faster than the ***Exact*** method. If high values of the filter parameter are used, the ***Fast*** method may produce a lot of noise. In this case, to reduce it, it is enough to increase the number of subdivisions, by using the next parameter.

filter subdivs - this parameter allows one to set the quality of the shadows when the ***Fast*** method is active. Low values produce a lot of noise, but with short rendering times. Vice versa, high values produce good-quality renders, but with long rendering times.

■ **Figure 9.136**
Notice how, as noise decreases, rendering time increases exponentially.

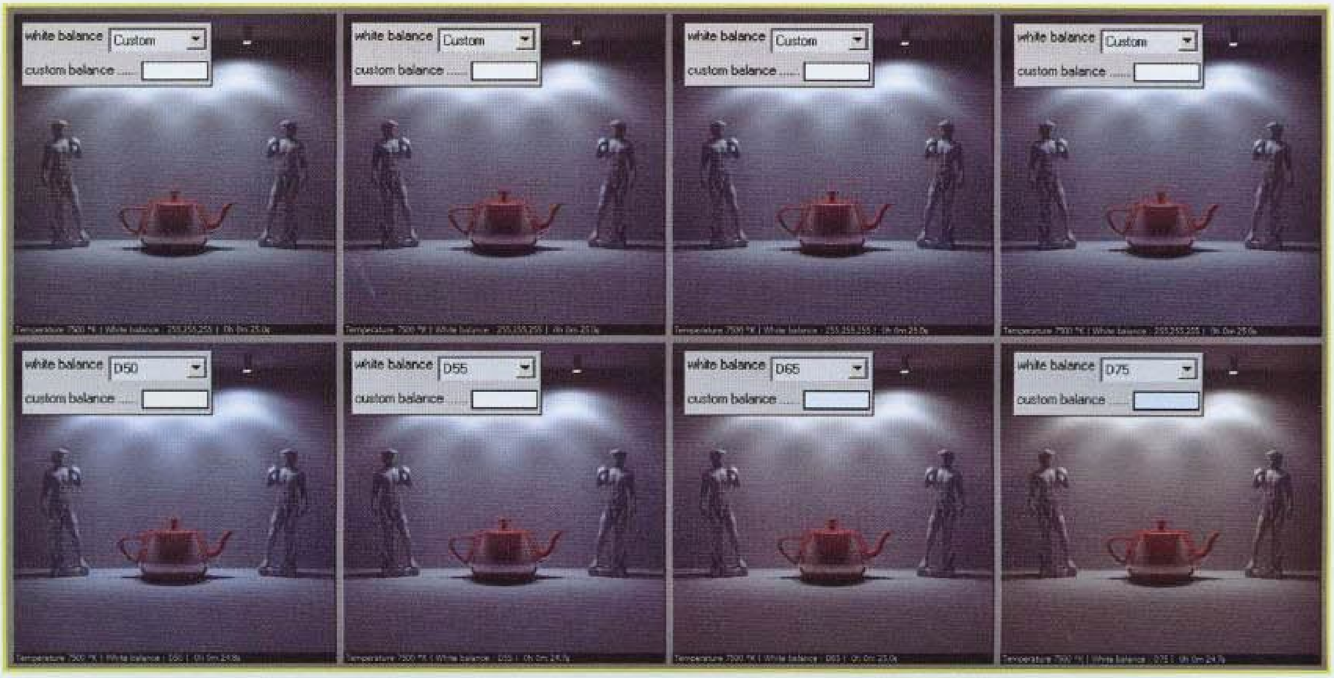
VRayPhysicalCamera

A new parameter, devoted to the counterbalance of white, has been introduced in the ***VRayPhisicalCamera***.



■ **Figure 9.137**
The new pull-down menu located in the ***VRayPhysicalCamera***.

Before seeing practical examples, a short digression on the counterbalance of white ensues. As has been seen in the previous pages, the new ***VRayIES*** (and probably future ***VRayLights*** as well), has the feature of expressing its color by means of the temperature. This way the scene acquires some color dominance, going from the orange for low temperatures, up to light blue for high ones. Balancing white means removing these dominants, in such a way that white is really white, and not orange or light blue.
Regarding this, the ***white balance*** parameter and its pull-down menu can be useful. For example, if a room is lit with a light with a temperature of **7500°K** without any counterbalance of white, the environment is going to appear very cold, because it tends towards light blue. If the counterbalance of the white is set to ***D75***, this dominant disappears and the scene will be balanced. Here below are some examples of comparisons between renders with and without the counterbalance and with new presets.



■ **Figure 9.138**
Different values of ***white balance***.

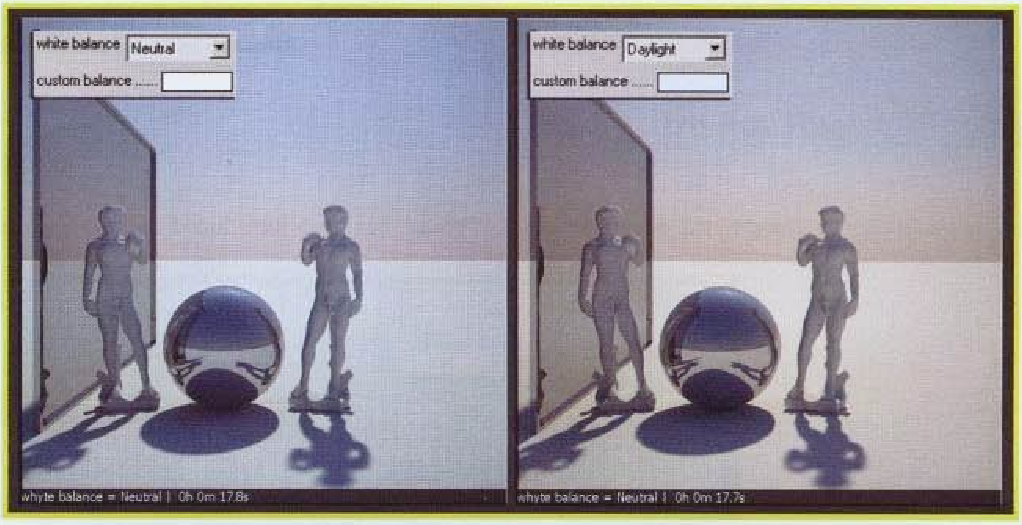
3 other presets are present: ***Custom***, ***Neutral*** and ***Daylight***.

Custom - this preset allows one to specify any color to be used for the white balance.

Neutral - the color to be used for the counterbalance is precisely RGB = [255, 255, 255].

Daylight - preset to be used together with the ***VRaySun*** + ***VRaySky*** system.

■ **Figure 9.139**
White balance in a daylight situation. By using the ***Daylight*** preset the light blue dominant tends to disappear, making the image better balanced.



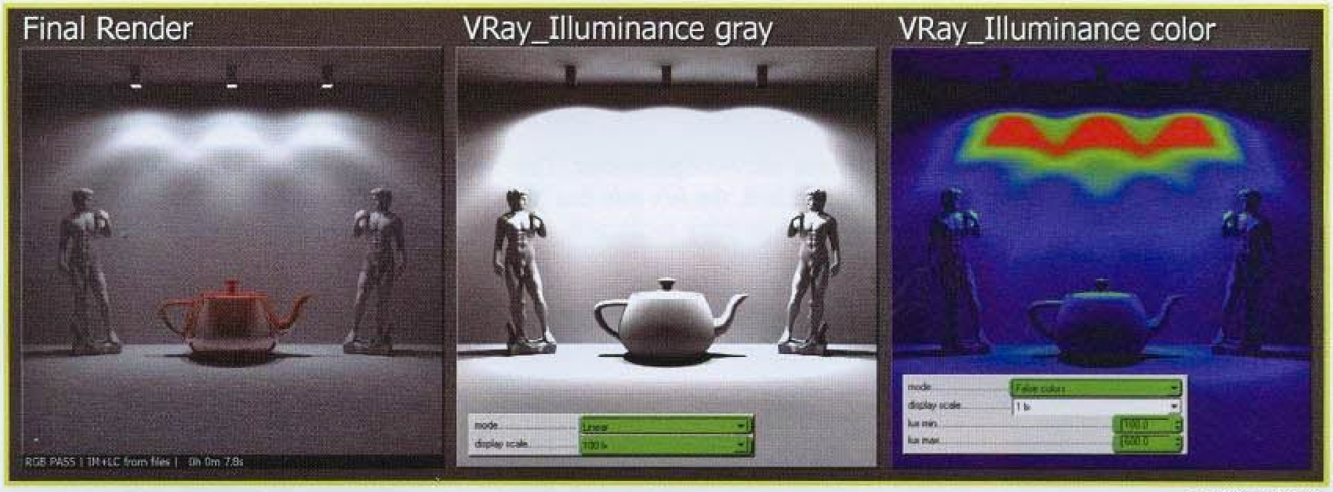
Render Elements

Chaosgroup has introduced 4 new Render Elements with *VRay 1.5 SP2*. Some were already acquired way back, such as the ***VRay_BumpNormals*** and the ***VRay_ExtraTex***, and others have been a pleasant surprise, such as the ***VRay_Illuminance*** and the ***VRay_SampleRate***.

VRay_Illuminance - this Render Element allows one to analyze the illumination engineering situation of the scene, thanks to the generation of a grey-scale or a gradient from red to blue render.

■ **Figure 9.140**
The interface of the Render Element ***VRay_Illuminance***.

Here below the Final render and the grey scale or colored ***VRay_Illuminance*** are represented.



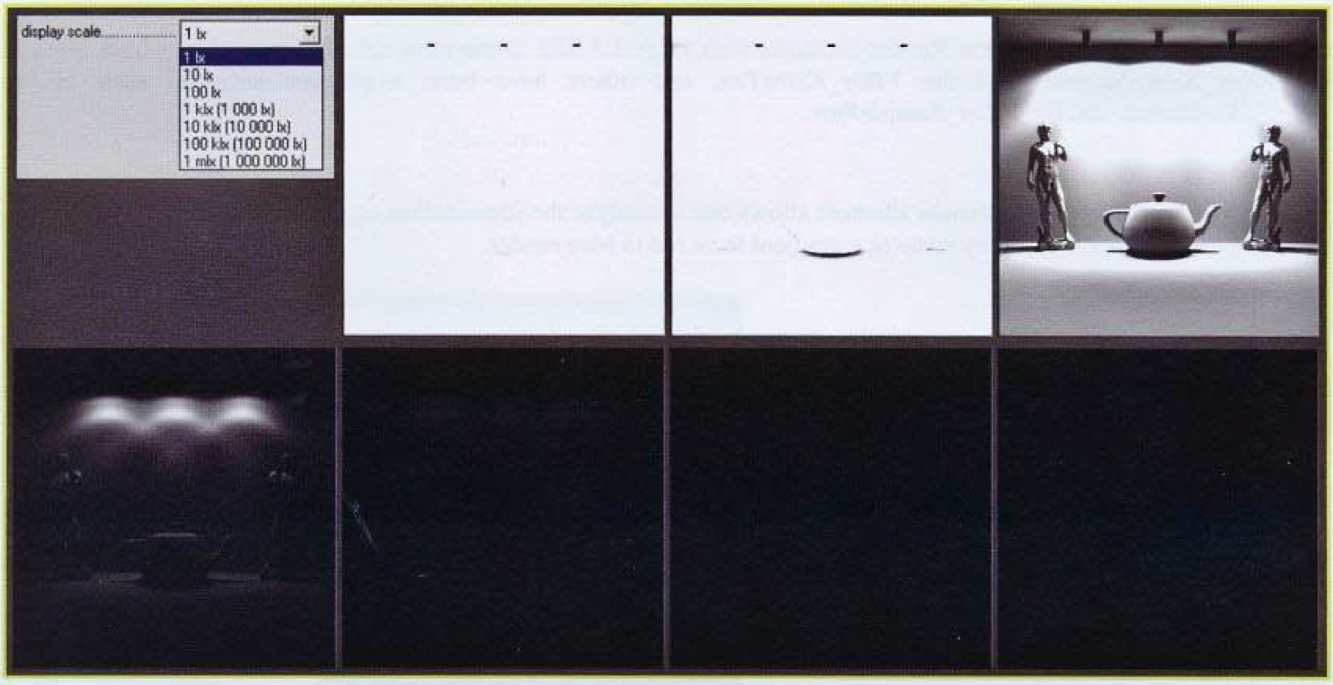
■ **Figure 9.141**
Some examples achieved thanks to the new Render Element.

mode - this parameter allows one to select the output mode, that is or the grey-scale or the gradient from the red to the blue.

■ **Figure 9.142**
The pull-down menu of the mode parameter.

By selecting ***Linear***, the ***lux min*** and ***lux max*** options are not considered, instead it is possible to operate on the Render Element by working exclusively on the ***display scale*** parameter. Vice versa, by selecting ***False colors***, it is possible to work on the colored gradient with the parameters which were previously obscured.

display scale - working on this pull-down menu it is possible to select the linear scale thanks to which *VRay* represents the lux present on the scene.



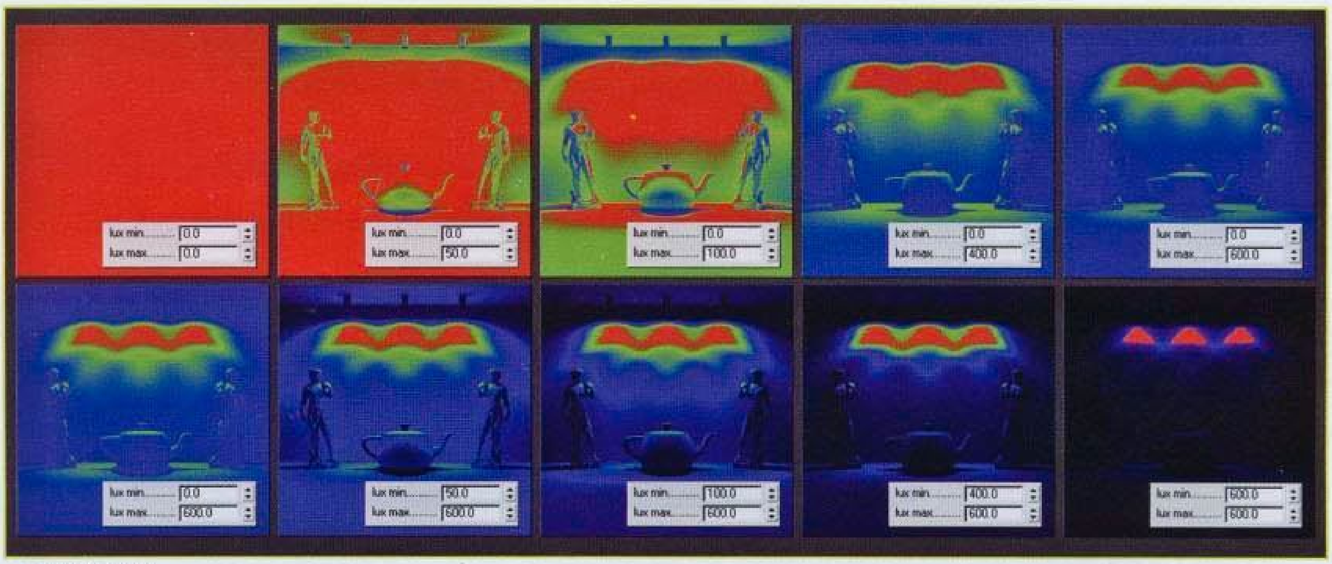
■ **Figure 9.143**
Render with different ***display scale*** values and by using the grey scale.

For sure the representation with the color gradient is stronger. In this case, to modify the Render Element, it is necessary to work on two parameters: ***lux min*** and ***lux max***.

lux min - this parameter defines the minimum value in lux represented by the dark blue color. The render will have a dark color if the light is equal to ***lux min***. Higher values tend to green.

lux max - this parameter defines the maximum value in lux represented by the red color. The render will have the color RGB = [255, 0, 0] if the light is equal to ***lux max***. Higher values are always red, and lower values tend to green first, and then to blue.

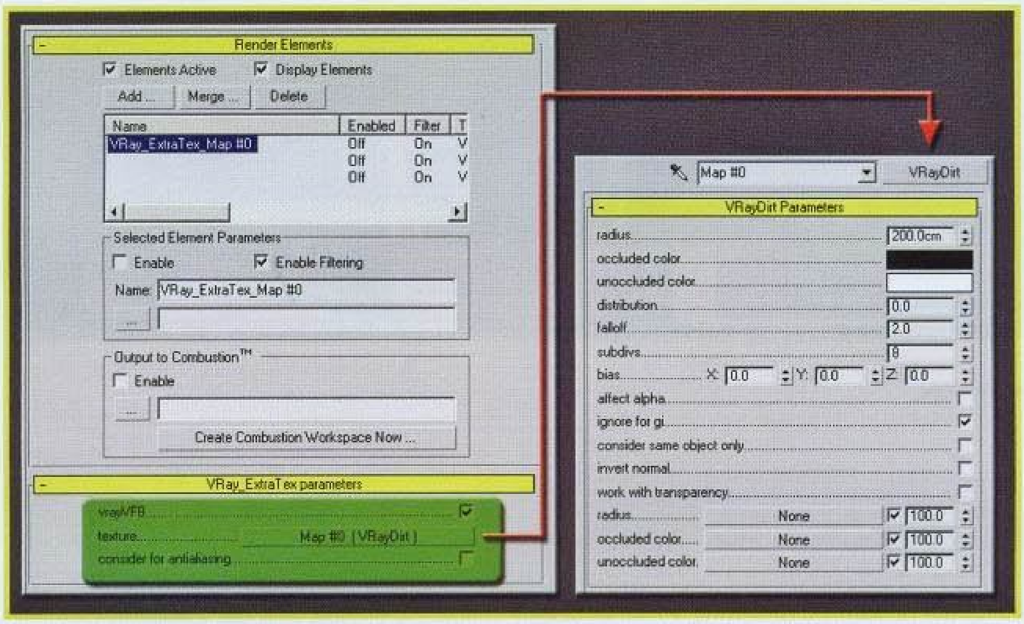
First of all set both values to 0. Then modify the ***lux max*** parameter. This way the gradient starts to appear, and the red starts to diminish. When the right coloring is reached, the ***lux min*** can be set. This way the minimum threshold is lifted, and the black color starts to appear as well. The last thing to do is to find the right equilibrium which, in the case here considered, seems to be ***lux min*** = 50 e ***lux max*** = 600.



■ **Figure 9.144**
Rendering with different ***display scale*** values and by using a colored gradient.

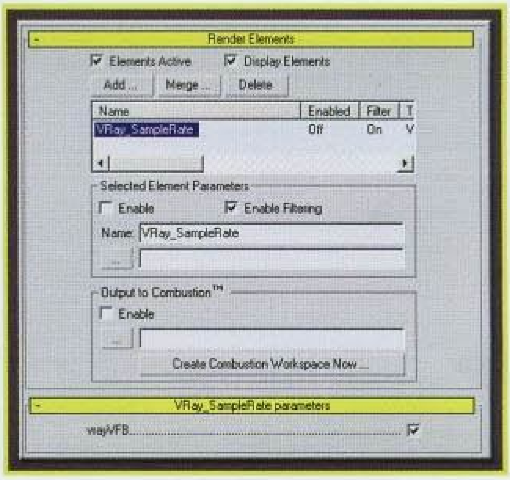
VRay_BumpNormal - this Render Element allows one to extract the bump channel within the Normal Pass. There is no parameter to select. It is enough to insert it in the Render Elements list to be generated and *VRay* looks after the computing of the channel.

VRay_ExtraTex - this Render Element allows one to render a new channel by specifying the texture to be used as a general map. This Pass has been studied mainly for computing the Ambient Occlusion or the ***VRayDirt*** together with the final render, so that one can use it with compositing programs, without having to launch the render one second time. In any case it is possible to use other textures, such as ***VRayEdgesTex***. Lastly, it is possible to compute multiple Render Elements at the same time, in such a way to produce multiple Passes simultaneously.



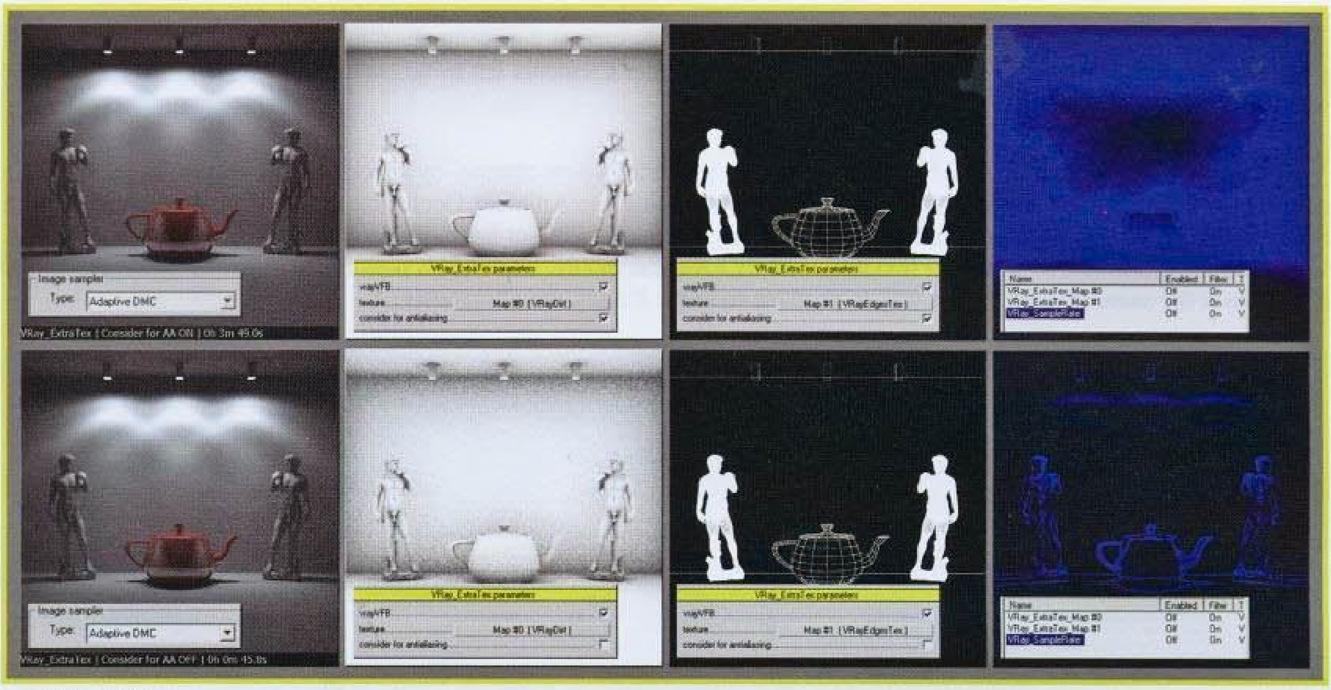
■ **Figure 9.145**
The new ***VRay_ExtraTex*** Render Element with the ***VRayDirt*** texture applied. The only available parameter is the ***consider for antialiasing***. By activating it, the new layer is subject to antialiasing, by using the sampling parameters applied to the final image. If it is deactivated, the ***VRay_ExtraTex*** is not subject to antialiasing, even if it is active on the final render.

VRay_SampleRate - this Render Element allows one to compute in a separate channel the samples used by *VRay* for the computing of the antialiasing. Essentially, it renders an image as if the ***Show samples*** parameter present in the ***VRay: Adaptive DMC image samples*** rollout is active. It is useful to analyze the behaviour of the antialiasing and verify possible problems.



■ **Figure 9.146**
There is no conFigureble parameter for this Render Element.

In the following example, in one step, 3 Render Elements have been rendered: 2 ***VRay_ExtraTex*** and ***VRay_SampleRate***. A ***VRayDirt*** and a ***VRayEdgesTex*** have been used as textures. Both of them with the ***consider for antialiasing*** option first activated and then deactivated. In these two configurations, the rendering time has passed from 3min. 49 sec. to 45sec., while as for the final render it has not changed. This evident increase of the time is due to the fact that the Pass of the AO has required many samples, and since the antialiasing for that Pass was active, *VRay* has been obliged to use a lot of time for the computing. This phenomenon is very clear in the ***VRay_SampleRate***, where the density of blue samples is very high, like when the ***consider for antialising*** is active.



■ **Figure 9.147**
Example of the use of the new Render Elements.

Profile MAX.vray

Finally Chaosgroup has provided the possibility of setting *VRay* as default rendering engine, and the ***VRayMtl*** as preset material instead of the Standard material. In such a way, each time 3ds Max is launched, everything is going to be ready to be rendered with *VRay*.

■ **Figure 9.148**
Settings for selecting ***VRay*** as the default working environment.